Block I
Apollo Guidance Computer (AGC)

How to build one in your basement

Part 9: Test and Checkout Software

John Pultorak
December, 2004

Abstract

This report describes my successful project to build a working reproduction of the 1964 prototype for the Block I Apollo Guidance Computer. The AGC is the flight computer for the Apollo moon landings, and is the world's first integrated circuit computer.

I built it in my basement. It took me 4 years.

If you like, you can build one too. It will take you less time, and yours will be better than mine.

I documented my project in 9 separate .pdf files:

Part 1 Overview: Introduces the project.

Part 2 CTL Module: Design and construction of the control module.

Part 3 PROC Module: Design and construction of the processing (CPU) module.

Part 4 MEM Module: Design and construction of the memory module.

Part 5 IO Module: Design and construction of the display/keyboard (DSKY) module.

Part 6 Assembler: A cross-assembler for AGC software development.

Part 7 C++ Simulator: A low-level simulator that runs assembled AGC code.

Part 8 Flight Software: My translation of portions of the COLOSSUS 249 flight software.

Part 9 Test & Checkout: A suite of test programs in AGC assembly language.

Overview

A suite of test and checkout programs were coded to verify the operation of the AGC simulators and the hardware AGC.

TECO1

First test and checkout program for the Block 1 AGC. Tests basic instructions: TC, CCS, INDEX, XCH, CS, TS, AD, MASK. Enters an infinite loop at the end of the test. The A register contains the code for the test that failed, or the PASS code if all tests succeeded. See test codes (in octal) below.

Code	Interpretation
01	TC check failed
02	CCS check failed
03	INDEX check failed
04	XCH check failed
05	CS check failed
06	TS check failed
07	AD check failed
10	MASK check failed
12345	PASSED all checks

TECO2

Second test and checkout program for the Block 1 AGC. Tests extracode instructions: MP, DV, SU. Enters an infinite loop at the end of the test. The A register contains the code for the test that failed, or the PASS code if all tests succeeded. See test codes (in octal) below.

Code	Interpretation
01	MP check failed
02	DV check failed
03	SU check failed
12345	PASSED all checks

TECO3

Third test and checkout program for the Block 1 AGC. Tests editing registers: CYR, SR, CYL, SL. Enters an infinite loop at the end of the test. The A register contains the code for the test that failed, or the PASS code if all tests succeeded. See test codes (in octal) below.

Code	Interpretation
01	CYR check failed
02	SR check failed
03	CYL check failed
04	SL check failed
12345	PASSED all checks

TECO5

Exercises AGC interrupts by initializing 4 counters, and then entering into a loop that increments the first counter on each iteration. Each of the other 3 counters is assigned to an interrupt and is incremented in the interrupt service routine for that interrupt: KEYRUPT, T3RUPT, and DSRUPT. Interrupts are inhibited and enabled during each iteration of the main loop with INHINT and RELINT instructions, and are automatically inhibited during part of each iteration by an overflow condition in register A.

TECO_STBY

An extremely simple program for testing the STANDBY function. STANDBY is disabled for 2 NOOP instructions and then is enabled. After that, the program infinitely loops (TC TRAP) with STANDBY enabled.

TECO1 assembler listing

```
Block I Apollo Guidance Computer (AGC4) assembler version 1.6 for EPROM

First pass: generate symbol table.
Second pass: generate object code.

; TECO1 (file:teco1.asm)
;
; Version:  1.0
; Author:   John Pultorak
; Date:     9/14/2001
;
; PURPOSE:
; Test and checkout program for the Block 1 Apollo Guidance Computer.
; Tests basic instructions: TC, CCS, INDEX, XCH, CS, TS, AD, MASK.
;
; OPERATION:
; Enters an infinite loop at the end of the test. The A register
; contains the code for the test that failed, or the PASS code if all
; tests succeeded. See test codes below.
;
; ERRATA:
; - Written for the AGC4R assembler. The assembler directives and
; syntax differ somewhat from the original AGC assembler.
; - The tests attempt to check all threads, but are not exhaustive.
;
; SOURCES:
; Information on the Block 1 architecture: instruction set, instruction
; sequences, registers, register transfers, control pulses, memory and
; memory addressing, I/O assignments, interrupts, and involuntary
; counters was obtained from:
;
;   A. Hopkins, R. Alonso, and H. Blair-Smith, "Logical Description
;          for the Apollo Guidance Computer (AGC4)", R-393,
;          MIT Instrumentation Laboratory, Cambridge, MA, Mar. 1963.
;
; Supplementary information was obtained from:
;
;   R. Alonso, J. H. Laning, Jr. and H. Blair-Smith, "Preliminary
;          MOD 3C Programmer's Manual", E-1077, MIT Instrumentation
;          Laboratory, Cambridge, MA, Nov. 1961.
;
;   B. I. Savage and A. Drake, "AGC4 Basic Training Manual, Volume I",
;          E-2052, MIT Instrumentation Laboratory, Cambridge,
;          MA, Jan. 1967.
;
;   E. C. Hall, "MIT's Role in Project Apollo, Volume III, Computer
;          Subsystem", R-700, MIT Charles Stark Draper Laboratory,
;          Cambridge, MA, Aug. 1972.
;
;   A. Hopkins, "Guidance Computer Design, Part VI", source unknown.
;
;   A. I. Green and J. J. Rocchio, "Keyboard and Display System Program
;          for AGC (Program Sunrise)", E-1574, MIT Instrumentation
;          Laboratory, Cambridge, MA, Aug. 1964.
;
;   E, C. Hall, "Journey to the Moon: The History of the Apollo
;          Guidance Computer", AIAA, Reston VA, 1996.
;

START          EQU      %00

TCtst          EQU      %01            ; TC check failed
CCStst         EQU      %02            ; CCS check failed
INDEXtst       EQU      %03            ; INDEX check failed
XCHtst         EQU      %04            ; XCH check failed
CStst          EQU      %05            ; CS check failed
TStst          EQU      %06            ; TS check failed
ADtst          EQU      %07            ; AD check failed
MASKtst        EQU      %10            ; MASK check failed

PASS           EQU      %12345         ; PASSED all checks
; -----------------------------------------------
```

```
                                         ORG      EXTENDER
05777    5777     47777 0                DS       %47777         ; needed for EXTEND

                          OVFCNTR        EQU      %00034         ; overflow counter

                          ; ----------------------------------------------
                          ; ERASEABLE MEMORY -- DATA SEGMENT

                                         ORG      %100           ; start of data area
00100    0100     00000 1 curtest        DS       START          ; current test
00101    0101     00000 1 savQ           DS       %0

                          ; CCS test
00102    0102     00000 1 CCSk           DS       %0

                          ; INDEX test
00103    0103     00000 1 INDXval        DS       0

                          ; XCH test
                          ; pre-set in erasable memory because we don't
                          ; want to use XCH to initialize them prior to testing XCH.
00104    0104     00000 1 XCHkP0         DS       +0
00105    0105     77777 0 XCHkM0         DS       -0
00106    0106     52525 1 XCHkalt1       DS       %52525         ; alternating bit pattern 1
00107    0107     25252 0 XCHkalt2       DS       %25252         ; alternating bit pattern 2

                          ; TS test
00110    0110     77777 0 TSk            DS       -0

                          ; AD test
00111    0111     77777 0 ADk            DS       -0

                          ; ----------------------------------------------
                          ; ENTRY POINTS

                          ; program (re)start
                                         ORG      GOPROG
02000    2000 0  1,2030 0                TC       goMAIN

                          ; interrupt service entry points
                                         ORG      T3RUPT
02004    2004 5  0,0026 0                TS       ARUPT
02005    2005 3  0,0001 0                XCH      Q
02006    2006 5  0,0027 1                TS       QRUPT
02007    2007 0  1,2717 1                TC       goT3

                                         ORG      ERRUPT
02010    2010 5  0,0026 0                TS       ARUPT
02011    2011 3  0,0001 0                XCH      Q
02012    2012 5  0,0027 1                TS       QRUPT
02013    2013 0  1,2717 1                TC       goER

                                         ORG      DSRUPT
02014    2014 5  0,0026 0                TS       ARUPT
02015    2015 3  0,0001 0                XCH      Q
02016    2016 5  0,0027 1                TS       QRUPT
02017    2017 0  1,2717 1                TC       goDS

                                         ORG      KEYRUPT
02020    2020 5  0,0026 0                TS       ARUPT
02021    2021 3  0,0001 0                XCH      Q
02022    2022 5  0,0027 1                TS       QRUPT
02023    2023 0  1,2717 1                TC       goKEY


                                         ORG      UPRUPT
02024    2024 5  0,0026 0                TS       ARUPT
02025    2025 3  0,0001 0                XCH      Q
02026    2026 5  0,0027 1                TS       QRUPT
02027    2027 0  1,2717 1                TC       goUP

                          ; ----------------------------------------------
                          ; FIXED MEMORY -- SHARED DATA SEGMENT

                          ; ----------------------------------------------
                          ; MAIN PROGRAM

                          goMAIN         EQU      *
02030    2030 2  0,0000 0                INHINT                  ; disable interrupts

02031    2031 0  1,2047 0                TCR      begin

                          ; Test basic instructions.
02032    2032 0  1,2054 1                TCR      chkTC
02033    2033 0  1,2110 0                TCR      chkCCS
02034    2034 0  1,2244 1                TCR      chkINDEX
02035    2035 0  1,2274 1                TCR      chkXCH
02036    2036 0  1,2400 1                TCR      chkCS
02037    2037 0  1,2446 0                TCR      chkTS
02040    2040 0  1,2573 1                TCR      chkAD
02041    2041 0  1,2674 0                TCR      chkMASK

                          ; Passed all tests.
02042    2042 0  1,2714 1                TCR      finish

                          fail           EQU      *
02043    2043 3  0,0100 0                XCH      curtest        ; load last passed test into A
02044    2044 5  0,0100 0                TS       curtest

                          end            EQU      *
02045    2045 0  1,2045 1                TC       end            ; finished, TC trap

                          ; ----------------------------------------------
                          ; INITIALIZE FOR START OF TESTING

02046    2046     00000 1 STRTcode       DS       START

                          begin          EQU      *
02047    2047 3  1,2046 1                XCH      STRTcode
02050    2050 5  0,0100 0                TS       curtest        ; set current test code to START
02051    2051 0  0,0000 0                RETURN

                          ; ----------------------------------------------
                          ; TEST TC INSTRUCTION SUBROUTINE
                          ; L:        TC     K
                          ; Verifies the following:
                          ; - Set C(Q) = TC L+1
                          ; - Take next instruction from K, and proceed from there.

02052    2052     00001 0 TCcode         DS       TCtst          ; code for this test
02053    2053     02061 1 Qtest          DS       TCret1         ; expected return address

                          chkTC          EQU      *
02054    2054 3  0,0001 0                XCH      Q
02055    2055 5  0,0101 1                TS       savQ           ; save return address

02056    2056 3  1,2052 1                CAF      TCcode
02057    2057 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; attempt a jump
02060    2060 0  1,2062 1                TC       *+2            ; make test jump
02061    2061 0  1,2043 1 TCret1         TC       fail           ; failed to jump

                          ; verify correct return address in Q
02062    2062 4  0,0001 1                CS       Q
02063    2063 6  1,2053 0                AD       Qtest          ; put (-Q) + val2 in A
02064    2064 1  0,0000 0                CCS      A              ; A = DABS
02065    2065 0  1,2043 1                TC       fail           ; >0 (Q < Qtest)
02066    2066 0  1,2043 1                TC       fail           ; +0 (never happens)
02067    2067 0  1,2043 1                TC       fail           ; <0 (Q > Qtest)

                          ; passed the test
02070    2070 3  0,0101 1                XCH      savQ
02071    2071 5  0,0001 0                TS       Q              ; restore return address
02072    2072 0  0,0000 0                RETURN
                          ; ----------------------------------------------
                          ; TEST CCS INSTRUCTION SUBROUTINE
                          ; L:        CCS    K
                          ; Verifies the following:
                          ; - take next instruction from L+n and proceed from there, where:
                          ; -- n = 1 if C(K) > 0
                          ; -- n = 2 if C(K) = +0
                          ; -- n = 3 if C(K) < 0
                          ; -- n = 4 if C(K) = -0
                          ; - set C(A) = DABS[C(K)], where DABS (diminished abs value):
                          ; -- DABS(a) = abs(a) - 1,        if abs(a) > 1
                          ; -- DABS(a) = +0,                if abs(a) <= 1

02073    2073     00002 0 CCScode        DS       CCStst         ; code for this test
                          ; test values (K)
02074    2074     77775 1 CCSkM2         DS       -2
02075    2075     77776 1 CCSkM1         DS       -1
02076    2076     77777 0 CCSkM0         DS       -0
02077    2077     00000 1 CCSkP0         DS       +0
02100    2100     00001 0 CCSkP1         DS       +1
02101    2101     00002 0 CCSkP2         DS       +2

                          ; expected DABS values
02102    2102     00001 0 CCSdM2         DS       1              ; for K=-2, DABS = +1
02103    2103     00000 1 CCSdM1         DS       0              ; for K=-1, DABS = +0
02104    2104     00000 1 CCSdM0         DS       0              ; for K=-0, DABS = +0
02105    2105     00000 1 CCSdP0         DS       0              ; for K=+0, DABS = +0
02106    2106     00000 1 CCSdP1         DS       0              ; for K=+1, DABS = +0
02107    2107     00001 0 CCSdP2         DS       1              ; for K=+2, DABS = +1

                          chkCCS         EQU      *
02110    2110 3  0,0001 0                XCH      Q
02111    2111 5  0,0101 1                TS       savQ           ; save return address

02112    2112 3  1,2073 1                CAF      CCScode
02113    2113 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; set K to -2 and execute CCS:
                          ; check for correct branch
02114    2114 3  1,2074 0                CAF      CCSkM2         ; set K = -2
02115    2115 5  0,0102 1                TS       CCSk
02116    2116 1  0,0102 0                CCS      CCSk           ; A = DABS[C(K)]
02117    2117 0  1,2043 1                TC       fail           ; K > 0
02120    2120 0  1,2043 1                TC       fail           ; K= +0
02121    2121 0  1,2123 0                TC       *+2            ; K < 0
02122    2122 0  1,2043 1                TC       fail           ; K= -0
                          ; check for correct DABS in A (for K=-2, it should be 1)
02123    2123 4  0,0000 0                COM                     ; 1's compliment of A
02124    2124 6  1,2102 0                AD       CCSdM2         ; put (-A) + expected value in A
02125    2125 1  0,0000 0                CCS      A              ; A = DABS
02126    2126 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02127    2127 0  1,2043 1                TC       fail           ; +0
02130    2130 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; set K to -1 and execute CCS:
                          ; check for correct branch
02131    2131 3  1,2075 1                CAF      CCSkM1         ; set K = -1
02132    2132 5  0,0102 1                TS       CCSk
02133    2133 1  0,0102 0                CCS      CCSk           ; A = DABS[C(K)]
02134    2134 0  1,2043 1                TC       fail           ; K > 0
02135    2135 0  1,2043 1                TC       fail           ; K= +0
02136    2136 0  1,2140 0                TC       *+2            ; K < 0
02137    2137 0  1,2043 1                TC       fail           ; K= -0
                          ; check for correct DABS in A (for K=-1, it should be +0)
02140    2140 4  0,0000 0                COM                     ; 1's compliment of A
02141    2141 6  1,2103 1                AD       CCSdM1         ; put (-A) + expected value in A
02142    2142 1  0,0000 0                CCS      A              ; A = DABS
02143    2143 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02144    2144 0  1,2043 1                TC       fail           ; +0
02145    2145 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; set K to -0 and execute CCS:
                          ; check for correct branch
02146    2146 3  1,2076 1                CAF      CCSkM0         ; set K = -0
02147    2147 5  0,0102 1                TS       CCSk
02150    2150 1  0,0102 0                CCS      CCSk           ; A = DABS[C(K)]
02151    2151 0  1,2043 1                TC       fail           ; K > 0
02152    2152 0  1,2043 1                TC       fail           ; K= +0
02153    2153 0  1,2043 1                TC       fail           ; K < 0
                          ; check for correct DABS in A (for K=-0, it should be +0)
02154    2154 4  0,0000 0                COM                     ; 1's compliment of A
02155    2155 6  1,2104 0                AD       CCSdM0         ; put (-A) + expected value in A
```

```
02156    2156 1  0,0000 0                CCS      A              ; A = DABS
02157    2157 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02160    2160 0  1,2043 1                TC       fail           ; +0
02161    2161 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; set K to +0 and execute CCS:
                          ; check for correct branch
02162    2162 3  1,2077 0                CAF      CCSkP0         ; set K = +0
02163    2163 5  0,0102 1                TS       CCSk
02164    2164 1  0,0102 0                CCS      CCSk           ; A = DABS[C(K)]
02165    2165 0  1,2043 1                TC       fail           ; K > 0
02166    2166 0  1,2171 1                TC       *+3            ; K= +0
02167    2167 0  1,2043 1                TC       fail           ; K < 0
02170    2170 0  1,2043 1                TC       fail           ; K= -0
                          ; check for correct DABS in A (for K=+0, it should be +0)
02171    2171 4  0,0000 0                COM                     ; 1's compliment of A
02172    2172 6  1,2105 1                AD       CCSdP0         ; put (-A) + expected value in A
02173    2173 1  0,0000 0                CCS      A              ; A = DABS
02174    2174 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02175    2175 0  1,2043 1                TC       fail           ; +0
02176    2176 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; set K to +1 and execute CCS:
                          ; check for correct branch
02177    2177 3  1,2100 1                CAF      CCSkP1         ; set K = +1
02200    2200 5  0,0102 1                TS       CCSk
02201    2201 1  0,0102 0                CCS      CCSk           ; A = DABS[C(K)]
02202    2202 0  1,2206 1                TC       *+4            ; K > 0
02203    2203 0  1,2043 1                TC       fail           ; K= +0
02204    2204 0  1,2043 1                TC       fail           ; K < 0
02205    2205 0  1,2043 1                TC       fail           ; K= -0
                          ; check for correct DABS in A (for K=+1, it should be +0)
02206    2206 4  0,0000 0                COM                     ; 1's compliment of A
02207    2207 6  1,2106 1                AD       CCSdP1         ; put (-A) + expected value in A
02210    2210 1  0,0000 0                CCS      A              ; A = DABS
02211    2211 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02212    2212 0  1,2043 1                TC       fail           ; +0
02213    2213 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; set K to +2 and execute CCS:
                          ; check for correct branch
02214    2214 3  1,2101 0                CAF      CCSkP2         ; set K = +2
02215    2215 5  0,0102 1                TS       CCSk
02216    2216 1  0,0102 0                CCS      CCSk           ; A = DABS[C(K)]
02217    2217 0  1,2223 0                TC       *+4            ; K > 0
02220    2220 0  1,2043 1                TC       fail           ; K= +0
02221    2221 0  1,2043 1                TC       fail           ; K < 0
02222    2222 0  1,2043 1                TC       fail           ; K= -0
                          ; check for correct DABS in A (for K=+2, it should be +1)
02223    2223 4  0,0000 0                COM                     ; 1's compliment of A
02224    2224 6  1,2107 0                AD       CCSdP2         ; put (-A) + expected value in A
02225    2225 1  0,0000 0                CCS      A              ; A = DABS
02226    2226 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02227    2227 0  1,2043 1                TC       fail           ; +0
02230    2230 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; passed the test
02231    2231 3  0,0101 1                XCH      savQ
02232    2232 5  0,0001 0                TS       Q              ; restore return address
02233    2233 0  0,0000 0                RETURN
                          ; ----------------------------------------------
                          ; TEST INDEX INSTRUCTION SUBROUTINE
                          ; L:       INDEX   K      (where K != 0025)
                          ; Verifies the following;
                          ; - Use the sum of C(L+1) + C(K) as the next instruction
                          ; -- just as if that sum had been taken from L+1.

02234    2234     00003 1 INDXcode       DS       INDEXtst       ; code for this test
02235    2235     00005 1 INDXst         DS       5              ; somewhere in fixed memory

02236    2236     00000 1 INDXbas        DS       0              ; base address for indexing
02237    2237     00001 0                DS       1
02240    2240     00002 0                DS       2
02241    2241     00003 1                DS       3
02242    2242     00004 0                DS       4
02243    2243     00005 1                DS       5
```

```
                          chkINDEX       EQU      *
02244    2244 3  0,0001 0                XCH      Q
02245    2245 5  0,0101 1                TS       savQ           ; save return address

02246    2246 3  1,2234 0                CAF      INDXcode
02247    2247 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; Decrementing loop
                          ;   - always executes at least once (tests at end of loop)
                          ;   - loops 'INDXst+1' times; decrements INDXval

02250    2250 3  1,2235 1                XCH      INDXst         ; initialize loop counter

                          INDXlop        EQU      *
02251    2251 5  0,0103 0                TS       INDXval

                          ; perform indexed CAF of values in INDXbas array;
                          ; index values range from 5 to 0
02252    2252 2  0,0103 1                INDEX    INDXval
02253    2253 3  1,2236 1                CAF      INDXbas

                          ; verify value retrieved using INDEX matches expected value
02254    2254 4  0,0000 0                COM                     ; get -A
02255    2255 6  0,0103 0                AD       INDXval        ; put (-A) + expected value in A
02256    2256 1  0,0000 0                CCS      A              ; compare
02257    2257 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02260    2260 0  1,2043 1                TC       fail           ; +0
02261    2261 0  1,2043 1                TC       fail           ; <0 (A > expected value)

02262    2262 1  0,0103 1                CCS      INDXval        ; done?
02263    2263 0  1,2251 0                TC       INDXlop        ; not yet

02264    2264 3  0,0101 1                XCH      savQ
02265    2265 5  0,0001 0                TS       Q              ; restore return address
02266    2266 0  0,0000 0                RETURN
                          ; ----------------------------------------------
                          ; TEST XCH INSTRUCTION SUBROUTINE
                          ; L:       XCH     K
                          ; Verifies the following:
                          ; - set C(A) = b(K)
                          ; - set C(K) = b(A)
                          ; - take next instruction from L+1

02267    2267     00004 0 XCHcode        DS       XCHtst         ; code for this test
                          ; XCH test values
02270    2270     00000 1 XCHfP0         DS       +0
02271    2271     77777 0 XCHfM0         DS       -0
02272    2272     52525 1 XCHfalt1       DS       %52525         ; alternating bit pattern 1
02273    2273     25252 0 XCHfalt2       DS       %25252         ; alternating bit pattern 2

                          chkXCH         EQU      *
02274    2274 3  0,0001 0                XCH      Q
02275    2275 5  0,0101 1                TS       savQ           ; save return address

02276    2276 3  1,2267 0                CAF      XCHcode
02277    2277 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; test - initial conditions: K=+0, A=-0
                          ; initialize A
02300    2300 4  1,2270 1                CS       XCHfP0
                          ; exchange A and K
02301    2301 3  0,0104 1                XCH      XCHkP0
                          ; test contents of A for expected value
02302    2302 4  0,0000 0                COM                     ; get -A
02303    2303 6  1,2270 0                AD       XCHfP0         ; put (-A) + expected value in A
02304    2304 1  0,0000 0                CCS      A              ; A = DABS
02305    2305 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02306    2306 0  1,2043 1                TC       fail           ; +0
02307    2307 0  1,2043 1                TC       fail           ; <0 (A > expected value)
                          ; test contents of K for expected value
02310    2310 4  0,0104 0                CS       XCHkP0         ; get -A
02311    2311 6  1,2271 1                AD       XCHfM0         ; put (-A) + expected value in A
02312    2312 1  0,0000 0                CCS      A              ; A = DABS
02313    2313 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02314    2314 0  1,2043 1                TC       fail           ; +0
```

```
02315    2315 0  1,2043 1               TC       fail           ; <0 (A > expected value)

                          ; test - initial conditions: K=-0, A=+0
                          ; initialize A
02316    2316 4  1,2271 0               CS       XCHfM0
                          ; exchange A and K
02317    2317 3  0,0105 0               XCH      XCHkM0
                          ; test contents of A for expected value
02320    2320 4  0,0000 0               COM                     ; get -A
02321    2321 6  1,2271 1               AD       XCHfM0         ; put (-A) + expected value in A
02322    2322 1  0,0000 0               CCS      A              ; A = DABS
02323    2323 0  1,2043 1               TC       fail           ; >0 (A < expected value)
02324    2324 0  1,2043 1               TC       fail           ; +0
02325    2325 0  1,2043 1               TC       fail           ; <0 (A > expected value)
                          ; test contents of K for expected value
02326    2326 4  0,0105 1               CS       XCHkM0         ; get -A
02327    2327 6  1,2270 0               AD       XCHfP0         ; put (-A) + expected value in A
02330    2330 1  0,0000 0               CCS      A              ; A = DABS
02331    2331 0  1,2043 1               TC       fail           ; >0 (A < expected value)
02332    2332 0  1,2043 1               TC       fail           ; +0
02333    2333 0  1,2043 1               TC       fail           ; <0 (A > expected value)

                          ; test - initial conditions: K=52525, A=25252
                          ; initialize A
02334    2334 4  1,2272 0               CS       XCHfalt1
                          ; exchange A and K
02335    2335 3  0,0106 0               XCH      XCHkalt1
                          ; test contents of A for expected value
02336    2336 4  0,0000 0               COM                     ; get -A
02337    2337 6  1,2272 1               AD       XCHfalt1       ; put (-A) + expected value in A
02340    2340 1  0,0000 0               CCS      A              ; A = DABS
02341    2341 0  1,2043 1               TC       fail           ; >0 (A < expected value)
02342    2342 0  1,2043 1               TC       fail           ; +0
02343    2343 0  1,2043 1               TC       fail           ; <0 (A > expected value)
                          ; test contents of K for expected value
02344    2344 4  0,0106 1               CS       XCHkalt1       ; get -A
02345    2345 6  1,2273 0               AD       XCHfalt2       ; put (-A) + expected value in A
02346    2346 1  0,0000 0               CCS      A              ; A = DABS
02347    2347 0  1,2043 1               TC       fail           ; >0 (A < expected value)
02350    2350 0  1,2043 1               TC       fail           ; +0
02351    2351 0  1,2043 1               TC       fail           ; <0 (A > expected value)

                          ; test - initial conditions: K=25252, A=52525
                          ; initialize A
02352    2352 4  1,2273 1               CS       XCHfalt2
                          ; exchange A and K
02353    2353 3  0,0107 1               XCH      XCHkalt2
                          ; test contents of A for expected value
02354    2354 4  0,0000 0               COM                     ; get -A
02355    2355 6  1,2273 0               AD       XCHfalt2       ; put (-A) + expected value in A
02356    2356 1  0,0000 0               CCS      A              ; A = DABS
02357    2357 0  1,2043 1               TC       fail           ; >0 (A < expected value)
02360    2360 0  1,2043 1               TC       fail           ; +0
02361    2361 0  1,2043 1               TC       fail           ; <0 (A > expected value)
                          ; test contents of K for expected value
02362    2362 4  0,0107 0               CS       XCHkalt2       ; get -A
02363    2363 6  1,2272 1               AD       XCHfalt1       ; put (-A) + expected value in A
02364    2364 1  0,0000 0               CCS      A              ; A = DABS
02365    2365 0  1,2043 1               TC       fail           ; >0 (A < expected value)
02366    2366 0  1,2043 1               TC       fail           ; +0
02367    2367 0  1,2043 1               TC       fail           ; <0 (A > expected value)

                          ; passed the test
02370    2370 3  0,0101 1               XCH      savQ
02371    2371 5  0,0001 0               TS       Q              ; restore return address
02372    2372 0  0,0000 0               RETURN

                          ; ----------------------------------------------
                          ; TEST CS INSTRUCTION SUBROUTINE
                          ; L:       CS      K
                          ; Verifies the following:
                          ; - Set C(A) = -C(K)
                          ; - Take next instruction from L+1

02373    2373     00005 1 CScode         DS       CStst          ; code for this test
                          ; test values (K)
```

```
02374    2374      00000 1 CSkP0          DS       +0
02375    2375      77777 0 CSkM0          DS       -0
02376    2376      52525 1 CSkalt1        DS       %52525         ; 1's C of CSkalt2
02377    2377      25252 0 CSkalt2        DS       %25252         ; 1's C of CSkalt1

                           chkCS          EQU      *
02400    2400 3  0,0001 0                 XCH      Q
02401    2401 5  0,0101 1                 TS       savQ           ; save return address

02402    2402 3  1,2373 1                 CAF      CScode
02403    2403 5  0,0100 0                 TS       curtest        ; set test code to this test

                           ; clear and subtract +0
02404    2404 4  1,2374 1                 CS       CSkP0          ; load 1's comp of K into A
02405    2405 6  1,2374 0                 AD       CSkP0          ; put (-A) + expected value in A
02406    2406 1  0,0000 0                 CCS      A              ; compare
02407    2407 0  1,2043 1                 TC       fail           ; >0 (A < expected value)
02410    2410 0  1,2043 1                 TC       fail           ; +0
02411    2411 0  1,2043 1                 TC       fail           ; <0 (A > expected value)

                           ; clear and subtract -0
02412    2412 4  1,2375 0                 CS       CSkM0          ; load 1's comp of K into A
02413    2413 6  1,2375 1                 AD       CSkM0          ; put (-A) + expected value in A
02414    2414 1  0,0000 0                 CCS      A              ; compare
02415    2415 0  1,2043 1                 TC       fail           ; >0 (A < expected value)
02416    2416 0  1,2043 1                 TC       fail           ; +0
02417    2417 0  1,2043 1                 TC       fail           ; <0 (A > expected value)

                           ; clear and subtract alternating bit pattern %52525
02420    2420 4  1,2376 0                 CS       CSkalt1        ; load 1's comp of K into A
02421    2421 6  1,2376 1                 AD       CSkalt1        ; put (-A) + expected value in A
02422    2422 1  0,0000 0                 CCS      A              ; compare
02423    2423 0  1,2043 1                 TC       fail           ; >0 (A < expected value)
02424    2424 0  1,2043 1                 TC       fail           ; +0
02425    2425 0  1,2043 1                 TC       fail           ; <0 (A > expected value)

                           ; clear and subtract alternating bit pattern %25252
02426    2426 4  1,2377 1                 CS       CSkalt2        ; load 1's comp of K into A
02427    2427 6  1,2377 0                 AD       CSkalt2        ; put (-A) + expected value in A
02430    2430 1  0,0000 0                 CCS      A              ; compare
02431    2431 0  1,2043 1                 TC       fail           ; >0 (A < expected value)
02432    2432 0  1,2043 1                 TC       fail           ; +0
02433    2433 0  1,2043 1                 TC       fail           ; <0 (A > expected value)

                           ; passed the test
02434    2434 3  0,0101 1                 XCH      savQ
02435    2435 5  0,0001 0                 TS       Q              ; restore return address
02436    2436 0  0,0000 0                 RETURN
                           ; ----------------------------------------------
                           ; TEST TS INSTRUCTION SUBROUTINE
                           ; L;       TS      K
                           ; Verifies the following:
                           ; - Set C(K) = b(A)
                           ; - If b(A) contains no overflow,
                           ; -- C(A) = b(A); take next instruction from L+1
                           ; - If b(A) has positive overflow, C(A) = 000001;
                           ; -- take next instruction from L+2
                           ; - If b(A) has negative overflow, C(A) = 177776;
                           ; -- take next instruction from L+2

02437    2437      00006 1 TScode         DS       TStst          ; code for this test
02440    2440      00001 0 TSone          DS       +1
02441    2441      00000 1 TSzero         DS       +0
02442    2442      77777 0 TSmzero        DS       -0
02443    2443      77776 1 TSmone         DS       -1
02444    2444      37777 1 TSkP1          DS       %37777         ; TEST1: largest + num w/no ovf
02445    2445      40000 0 TSkM1          DS       %40000         ; TEST2: largest - num w/no ovf

                           chkTS          EQU      *
02446    2446 3  0,0001 0                 XCH      Q
02447    2447 5  0,0101 1                 TS       savQ           ; save return address

02450    2450 3  1,2437 0                 CAF      TScode
02451    2451 5  0,0100 0                 TS       curtest        ; set test code to this test

                           ; initialize TSk to -0
```

```
02452    2452 3  1,2442 1                CAF      TSmzero
02453    2453 3  0,0110 1                XCH      TSk

                          ; TEST 1: store positive number, no overflow
02454    2454 3  1,2444 1                CAF      TSkP1
02455    2455 5  0,0110 1                TS       TSk
02456    2456 0  1,2460 1                TC       *+2            ; no overflow
02457    2457 0  1,2043 1                TC       fail           ; overflow
                          ; verify C(A) = b(A)
02460    2460 4  0,0000 0                COM                     ; get -A
02461    2461 6  1,2444 1                AD       TSkP1          ; put (-A) + expected value in A
02462    2462 1  0,0000 0                CCS      A              ; compare
02463    2463 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02464    2464 0  1,2043 1                TC       fail           ; +0
02465    2465 0  1,2043 1                TC       fail           ; <0 (A > expected value)
                          ; verify C(K) = b(A)
02466    2466 4  1,2444 0                CS       TSkP1          ; get -expected value
02467    2467 6  0,0110 1                AD       TSk            ; put value + C(K) into A
02470    2470 1  0,0000 0                CCS      A              ; compare
02471    2471 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02472    2472 0  1,2043 1                TC       fail           ; +0
02473    2473 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; TEST 2: store negative number, no overflow
02474    2474 3  1,2445 0                CAF      TSkM1
02475    2475 5  0,0110 1                TS       TSk
02476    2476 0  1,2500 0                TC       *+2            ; no overflow
02477    2477 0  1,2043 1                TC       fail           ; overflow
                          ; verify C(A) = b(A)
02500    2500 4  0,0000 0                COM                     ; get -A
02501    2501 6  1,2445 0                AD       TSkM1          ; put (-A) + expected value in A
02502    2502 1  0,0000 0                CCS      A              ; compare
02503    2503 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02504    2504 0  1,2043 1                TC       fail           ; +0
02505    2505 0  1,2043 1                TC       fail           ; <0 (A > expected value)
                          ; verify C(K) = b(A)
02506    2506 4  1,2445 1                CS       TSkM1          ; get -expected value
02507    2507 6  0,0110 1                AD       TSk            ; put value + C(K) into A
02510    2510 1  0,0000 0                CCS      A              ; compare
02511    2511 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02512    2512 0  1,2043 1                TC       fail           ; +0
02513    2513 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; TEST 3: store positive number, overflow
02514    2514 3  1,2444 1                CAF      TSkP1          ; get largest positive number
02515    2515 6  1,2440 0                AD       TSone          ; make it overflow; A = neg ovf
02516    2516 5  0,0110 1                TS       TSk            ; store the positive overflow
02517    2517 0  1,2043 1                TC       fail           ; no overflow
                          ; verify C(A) = 000001
02520    2520 4  0,0000 0                COM                     ; get -A
02521    2521 6  1,2440 0                AD       TSone          ; put (-A) + expected value in A
02522    2522 1  0,0000 0                CCS      A              ; compare
02523    2523 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02524    2524 0  1,2043 1                TC       fail           ; +0
02525    2525 0  1,2043 1                TC       fail           ; <0 (A > expected value)
                          ; verify C(K) = positive overflow
02526    2526 4  1,2441 0                CS       TSzero         ; get -expected value
02527    2527 6  0,0110 1                AD       TSk            ; put value + C(K) into A
02530    2530 1  0,0000 0                CCS      A              ; compare
02531    2531 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02532    2532 0  1,2043 1                TC       fail           ; +0
02533    2533 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; TEST 4: store negative number, overflow
02534    2534 3  1,2445 0                CAF      TSkM1          ; get largest negative number
02535    2535 6  1,2443 0                AD       TSmone         ; make it overflow; A = neg ovf
02536    2536 5  0,0110 1                TS       TSk            ; store the negative overflow
02537    2537 0  1,2043 1                TC       fail           ; no overflow
                          ; verify C(A) = 177776
02540    2540 4  0,0000 0                COM                     ; get -A
02541    2541 6  1,2443 0                AD       TSmone         ; put (-A) + expected value in A
02542    2542 1  0,0000 0                CCS      A              ; compare
02543    2543 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02544    2544 0  1,2043 1                TC       fail           ; +0
02545    2545 0  1,2043 1                TC       fail           ; <0 (A > expected value)
                          ; verify C(K) = negative overflow
```

```
02546    2546 4  1,2442 0                CS       TSmzero         ; get -expected value
02547    2547 6  0,0110 1                AD       TSk             ; put value + C(K) into A
02550    2550 1  0,0000 0                CCS      A               ; compare
02551    2551 0  1,2043 1                TC       fail            ; >0 (A < expected value)
02552    2552 0  1,2043 1                TC       fail            ; +0
02553    2553 0  1,2043 1                TC       fail            ; <0 (A > expected value)

02554    2554 3  0,0101 1                XCH      savQ
02555    2555 5  0,0001 0                TS       Q               ; restore return address
02556    2556 0  0,0000 0                RETURN
                          ; ----------------------------------------------
                          ; TEST AD INSTRUCTION SUBROUTINE
                          ; L:       AD      K
                          ; Verifies the following:
                          ; - Set C(A) = b(A) + C(K)
                          ; - Take next instruction from L+1
                          ; - if C(A) has positive overflow,
                          ; -- increment overflow counter by 1
                          ; - if C(A) has negative overflow,
                          ; -- decrement overflow counter by 1

02557    2557     00007 0 ADcode         DS       ADtst           ; code for this test
02560    2560     00000 1 ADplus0        DS       +0
02561    2561     00001 0 ADplus1        DS       1
02562    2562     77776 1 ADmin1         DS       -1

02563    2563     25252 0 AD25252        DS       %25252          ; +10922 decimal
02564    2564     12525 0 AD12525        DS       %12525          ; +5461 decimal
02565    2565     37777 1 AD37777        DS       %37777          ; largest positive number
02566    2566     12524 1 AD12524        DS       %12524          ; + overflow of %25252+%25252

02567    2567     52525 1 AD52525        DS       %52525          ; -10922 decimal
02570    2570     65252 1 AD65252        DS       %65252          ; -5461 decimal
02571    2571     40000 0 AD40000        DS       %40000          ; largest negative number
02572    2572     65253 0 AD65253        DS       %65253          ; neg overflow of %52525+65252

                          chkAD          EQU      *
02573    2573 3  0,0001 0                XCH      Q
02574    2574 5  0,0101 1                TS       savQ            ; save return address

02575    2575 3  1,2557 1                CAF      ADcode
02576    2576 5  0,0100 0                TS       curtest         ; set test code to this test

                          ; TEST1: sum positive, no overflow
                          ; add: %25252 + %12525 = %37777 (sign + 14 magnitude)
02577    2577 3  1,2563 0                CAF      AD25252
02600    2600 6  1,2564 1                AD       AD12525
                          ; verify C(A) = %37777
02601    2601 4  0,0000 0                COM                      ; get -A
02602    2602 6  1,2565 0                AD       AD37777         ; put (-A) + expected value in A
02603    2603 1  0,0000 0                CCS      A               ; compare
02604    2604 0  1,2043 1                TC       fail            ; >0 (A < expected value)
02605    2605 0  1,2043 1                TC       fail            ; +0
02606    2606 0  1,2043 1                TC       fail            ; <0 (A > expected value)

                          ; TEST2: sum negative, no overflow (sign + 14 magnitude)
                          ; add: %52525 + %65252 = %40000
02607    2607 3  1,2567 1                CAF      AD52525
02610    2610 6  1,2570 1                AD       AD65252
                          ; verify C(A) = %40000
02611    2611 4  0,0000 0                COM                      ; get -A
02612    2612 6  1,2571 0                AD       AD40000         ; put (-A) + expected value in A
02613    2613 1  0,0000 0                CCS      A               ; compare
02614    2614 0  1,2043 1                TC       fail            ; >0 (A < expected value)
02615    2615 0  1,2043 1                TC       fail            ; +0
02616    2616 0  1,2043 1                TC       fail            ; <0 (A > expected value)

                          ; TEST3: sum positive, overflow
                          ; initialize overflow counter and positive overflow storage
02617    2617 3  1,2560 0                CAF      ADplus0
02620    2620 5  0,0034 0                TS       OVFCNTR
02621    2621 5  0,0111 0                TS       ADk
                          ; add: %25252 + %25252 = %52524 (sign + 14 magnitude)
02622    2622 3  1,2563 0                CAF      AD25252
02623    2623 6  1,2563 0                AD       AD25252
02624    2624 5  0,0111 0                TS       ADk             ; store positive overflow
```

```
02625    2625 0  1,2043 1                TC       fail
                          ; verify ADk = %12524
02626    2626 4  0,0111 1                CS       ADk            ; get -A
02627    2627 6  1,2566 0                AD       AD12524        ; put (-A) + expected value in A
02630    2630 1  0,0000 0                CCS      A              ; compare
02631    2631 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02632    2632 0  1,2043 1                TC       fail           ; +0
02633    2633 0  1,2043 1                TC       fail           ; <0 (A > expected value)
                          ; verify overflow counter =%00001
02634    2634 4  0,0034 1                CS       OVFCNTR        ; get -A
02635    2635 6  1,2561 1                AD       ADplus1        ; put (-A) + expected value in A
02636    2636 1  0,0000 0                CCS      A              ; compare
02637    2637 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02640    2640 0  1,2043 1                TC       fail           ; +0
02641    2641 0  1,2043 1                TC       fail           ; <0 (A > expected value)

                          ; TEST4: sum negative, overflow
02642    2642 3  1,2560 0                CAF      ADplus0
02643    2643 5  0,0034 0                TS       OVFCNTR
02644    2644 5  0,0111 0                TS       ADk
                          ; add: %52525 + %52525 = %25253 (sign + 14 magnitude)
02645    2645 3  1,2567 1                CAF      AD52525
02646    2646 6  1,2567 1                AD       AD52525
02647    2647 5  0,0111 0                TS       ADk            ; store negative overflow
02650    2650 0  1,2043 1                TC       fail
                          ; verify ADk = %65253
02651    2651 4  0,0111 1                CS       ADk            ; get -A
02652    2652 6  1,2572 0                AD       AD65253        ; put (-A) + expected value in A
02653    2653 1  0,0000 0                CCS      A              ; compare
02654    2654 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02655    2655 0  1,2043 1                TC       fail           ; +0
02656    2656 0  1,2043 1                TC       fail           ; <0 (A > expected value)
                          ; verify overflow counter =%77776
02657    2657 4  0,0034 1                CS       OVFCNTR        ; get -A
02660    2660 6  1,2562 1                AD       ADmin1         ; put (-A) + expected value in A
02661    2661 1  0,0000 0                CCS      A              ; compare
02662    2662 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02663    2663 0  1,2043 1                TC       fail           ; +0
02664    2664 0  1,2043 1                TC       fail           ; <0 (A > expected value)

02665    2665 3  0,0101 1                XCH      savQ
02666    2666 5  0,0001 0                TS       Q              ; restore return address
02667    2667 0  0,0000 0                RETURN
                          ; ----------------------------------------------
                          ; TEST MASK INSTRUCTION SUBROUTINE
                          ; L:        MASK    K
                          ; Verifies the following:
                          ; - Set C(A) = b(A) & C(K)

02670    2670      00010 0 MASKcode       DS       MASKtst        ; code for this test
02671    2671      46314 0 MASK1          DS       %46314
02672    2672      25252 0 MASK2          DS       %25252
02673    2673      04210 0 MASKval        DS       %04210         ; expected result: MASK1 & MASK2

                          chkMASK        EQU      *
02674    2674 3  0,0001 0                XCH      Q
02675    2675 5  0,0101 1                TS       savQ           ; save return address

                          ; perform logical and of MASK1 and MASK2
02676    2676 3  1,2671 0                CAF      MASK1
02677    2677 7  1,2672 1                MASK     MASK2
                          ; verify C(A) = b(A) & C(K)
02700    2700 4  0,0000 0                COM                     ; get -A
02701    2701 6  1,2673 1                AD       MASKval        ; put (-A) + expected value in A
02702    2702 1  0,0000 0                CCS      A              ; compare
02703    2703 0  1,2043 1                TC       fail           ; >0 (A < expected value)
02704    2704 0  1,2043 1                TC       fail           ; +0
02705    2705 0  1,2043 1                TC       fail           ; <0 (A > expected value)

02706    2706 3  1,2670 1                CAF      MASKcode
02707    2707 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; passed the test
02710    2710 3  0,0101 1                XCH      savQ
02711    2711 5  0,0001 0                TS       Q              ; restore return address
02712    2712 0  0,0000 0                RETURN
```

```
                          ; ---------------------------------------------
                          ; PASSED ALL TESTS!

02713    2713     12345 0 PASScode       DS       PASS

                          finish         EQU      *
02714    2714 3  1,2713 0                CAF      PASScode
02715    2715 5  0,0100 0                TS       curtest        ; set current test code to PASS
02716    2716 0  0,0000 0                RETURN

                          ; ---------------------------------------------
                          ; INTERRUPT SERVICE ROUTINE

                          goT3           EQU      *
                          goER           EQU      *
                          goDS           EQU      *
                          goKEY          EQU      *
                          goUP           EQU      *


                          endRUPT        EQU      *
02717    2717 3  0,0027 1                XCH      QRUPT          ; restore Q
02720    2720 5  0,0001 0                TS       Q
02721    2721 3  0,0026 0                XCH      ARUPT          ; restore A
02722    2722 2  0,0000 1                RESUME                  ; finished, go back



Assembly complete. Errors = 0

Symbol table:
START          000000   TCtst          000001   CCStst         000002
INDEXtst       000003   XCHtst         000004   CStst          000005
TStst          000006   ADtst          000007   MASKtst        000010
PASS           012345   EXTENDER       005777   OVFCNTR        000034
curtest        000100   savQ           000101   CCSk           000102
INDXval        000103   XCHkP0         000104   XCHkM0         000105
XCHkalt1       000106   XCHkalt2       000107   TSk            000110
ADk            000111   GOPROG         002000   T3RUPT         002004
ERRUPT         002010   DSRUPT         002014   KEYRUPT        002020
UPRUPT         002024   goMAIN         002030   fail           002043
end            002045   STRTcode       002046   begin          002047
TCcode         002052   Qtest          002053   chkTC          002054
TCret1         002061   CCScode        002073   CCSkM2         002074
CCSkM1         002075   CCSkM0         002076   CCSkP0         002077
CCSkP1         002100   CCSkP2         002101   CCSdM2         002102
CCSdM1         002103   CCSdM0         002104   CCSdP0         002105
CCSdP1         002106   CCSdP2         002107   chkCCS         002110
INDXcode       002234   INDXst         002235   INDXbas        002236
chkINDEX       002244   INDXlop        002251   XCHcode        002267
XCHfP0         002270   XCHfM0         002271   XCHfalt1       002272
XCHfalt2       002273   chkXCH         002274   CScode         002373
CSkP0          002374   CSkM0          002375   CSkalt1        002376
CSkalt2        002377   chkCS          002400   TScode         002437
TSone          002440   TSzero         002441   TSmzero        002442
TSmone         002443   TSkP1          002444   TSkM1          002445
chkTS          002446   ADcode         002557   ADplus0        002560
ADplus1        002561   ADmin1         002562   AD25252        002563
AD12525        002564   AD37777        002565   AD12524        002566
AD52525        002567   AD65252        002570   AD40000        002571
AD65253        002572   chkAD          002573   MASKcode       002670
MASK1          002671   MASK2          002672   MASKval        002673
chkMASK        002674   PASScode       002713   finish         002714
goT3           002717   goER           002717   goDS           002717
goKEY          002717   goUP           002717   endRUPT        002717
ARUPT          000026   Q              000001   QRUPT          000027
A              000000
```

TECO2 assembler listing

```
Block I Apollo Guidance Computer (AGC4) assembler version 1.6 for EPROM

First pass: generate symbol table.
Second pass: generate object code.

                          ; TECO2 (file:teco2.asm)
                          ;
                          ; Version:  1.0
                          ; Author:   John Pultorak
                          ; Date:     9/14/2001
                          ;
                          ; PURPOSE:
                          ; Test and checkout program for the Block 1 Apollo Guidance Computer.
                          ; Tests extracode instructions: MP, DV, SU
                          ;
                          ; OPERATION:
                          ; Enters an infinite loop at the end of the test. The A register
                          ; contains the code for the test that failed, or the PASS code if all
                          ; tests succeeded. See test codes below.
                          ;
                          ; ERRATA:
                          ; - Written for the AGC4R assembler. The assembler directives and
                          ; syntax differ somewhat from the original AGC assembler.
                          ; - The tests attempt to check all threads, but are not exhaustive.
                          ;
                          ; SOURCES:
                          ; Information on the Block 1 architecture: instruction set, instruction
                          ; sequences, registers, register transfers, control pulses, memory and
                          ; memory addressing, I/O assignments, interrupts, and involuntary
                          ; counters was obtained from:
                          ;
                          ;   A. Hopkins, R. Alonso, and H. Blair-Smith, "Logical Description
                          ;          for the Apollo Guidance Computer (AGC4)", R-393,
                          ;          MIT Instrumentation Laboratory, Cambridge, MA, Mar. 1963.
                          ;
                          ; Supplementary information was obtained from:
                          ;
                          ;   R. Alonso, J. H. Laning, Jr. and H. Blair-Smith, "Preliminary
                          ;          MOD 3C Programmer's Manual", E-1077, MIT Instrumentation
                          ;          Laboratory, Cambridge, MA, Nov. 1961.
                          ;
                          ;   B. I. Savage and A. Drake, "AGC4 Basic Training Manual, Volume I",
                          ;          E-2052, MIT Instrumentation Laboratory, Cambridge,
                          ;          MA, Jan. 1967.
                          ;
                          ;   E. C. Hall, "MIT's Role in Project Apollo, Volume III, Computer
                          ;          Subsystem", R-700, MIT Charles Stark Draper Laboratory,
                          ;          Cambridge, MA, Aug. 1972.
                          ;
                          ;   A. Hopkins, "Guidance Computer Design, Part VI", source unknown.
                          ;
                          ;   A. I. Green and J. J. Rocchio, "Keyboard and Display System Program
                          ;          for AGC (Program Sunrise)", E-1574, MIT Instrumentation
                          ;          Laboratory, Cambridge, MA, Aug. 1964.
                          ;
                          ;   E, C. Hall, "Journey to the Moon: The History of the Apollo
                          ;          Guidance Computer", AIAA, Reston VA, 1996.
                          ;


                          START          EQU      %00

                          MPtst          EQU      %01            ; MP check failed
                          DVtst          EQU      %02            ; DV check failed
                          SUtst          EQU      %03            ; SU check failed

                          PASS           EQU      %12345         ; PASSED all checks
                          ; ----------------------------------------------

                                         ORG      EXTENDER
05777    5777     47777 0                DS       %47777         ; needed for EXTEND
```

```
                          OVFCNTR        EQU      %00034         ; overflow counter

                          ; -----------------------------------------------
                          ; ERASEABLE MEMORY -- DATA SEGMENT

                                         ORG      %100           ; start of data area
00100    0100     00000 1 curtest        DS       START          ; current test
00101    0101     00000 1 savQ           DS       %0

                          ; MP test
00102    0102     00000 1 MPindex        DS       %0
00103    0103     00000 1 MPXTND         DS       %0             ; indexed extend

                          ; DV test
00104    0104     00000 1 DVsavA         DS       %0
00105    0105     00000 1 DVindex        DS       %0
00106    0106     00000 1 DVXTND         DS       %0             ; indexed extend

                          ; SU test
00107    0107     77777 0 SUk            DS       -0

                          ; -----------------------------------------------
                          ; ENTRY POINTS

                          ; program (re)start
                                         ORG      GOPROG
02000    2000 0  1,2030 0                TC       goMAIN

                          ; interrupt service entry points
                                         ORG      T3RUPT
02004    2004 5  0,0026 0                TS       ARUPT
02005    2005 3  0,0001 0                XCH      Q
02006    2006 5  0,0027 1                TS       QRUPT
02007    2007 0  1,2742 1                TC       goT3

                                         ORG      ERRUPT
02010    2010 5  0,0026 0                TS       ARUPT
02011    2011 3  0,0001 0                XCH      Q
02012    2012 5  0,0027 1                TS       QRUPT
02013    2013 0  1,2742 1                TC       goER

                                         ORG      DSRUPT
02014    2014 5  0,0026 0                TS       ARUPT
02015    2015 3  0,0001 0                XCH      Q
02016    2016 5  0,0027 1                TS       QRUPT
02017    2017 0  1,2742 1                TC       goDS

                                         ORG      KEYRUPT
02020    2020 5  0,0026 0                TS       ARUPT
02021    2021 3  0,0001 0                XCH      Q
02022    2022 5  0,0027 1                TS       QRUPT
02023    2023 0  1,2742 1                TC       goKEY


                                         ORG      UPRUPT
02024    2024 5  0,0026 0                TS       ARUPT
02025    2025 3  0,0001 0                XCH      Q
02026    2026 5  0,0027 1                TS       QRUPT
02027    2027 0  1,2742 1                TC       goUP


                          ; -----------------------------------------------
                          ; FIXED MEMORY -- SHARED DATA SEGMENT

                          ; -----------------------------------------------
                          ; MAIN PROGRAM

                          goMAIN         EQU      *
02030    2030 2  0,0000 0                INHINT                  ; disable interrupts

02031    2031 0  1,2042 0                TCR      begin

                          ; Test extracode instructions.
02032    2032 0  1,2247 1                TCR      chkMP
02033    2033 0  1,2551 1                TCR      chkDV
02034    2034 0  1,2635 0                TCR      chkSU
```

```
                          ; Passed all tests.
02035    2035 0  1,2737 0                TCR      finish

                          fail           EQU      *
02036    2036 3  0,0100 0                XCH      curtest        ; load last passed test into A
02037    2037 5  0,0100 0                TS       curtest

                          end            EQU      *
02040    2040 0  1,2040 1                TC       end            ; finished, TC trap

                          ; ----------------------------------------------
                          ; INITIALIZE FOR START OF TESTING

02041    2041      00000 1 STRTcode       DS       START

                          begin          EQU      *
02042    2042 3  1,2041 0                XCH      STRTcode
02043    2043 5  0,0100 0                TS       curtest        ; set current test code to START
02044    2044 0  0,0000 0                RETURN

                          ; ----------------------------------------------
                          ; TEST MP INSTRUCTION SUBROUTINE
                          ; L:        MP     K
                          ; Verifies the following
                          ; - Set C(A,LP) = b(A) * C(K)
                          ; - Take next instruction from L+1

02045    2045      00001 0 MPcode         DS       MPtst          ; code for this test

                          ; MP test values
                          ;
02046    2046      00037 0 MPstart        DS       31             ; loop MPstart+1 times

                          ; C(A) test values
                          mp1            EQU      *
                          ; check boundary conditions
02047    2047      37777 1                DS       %37777         ; check #00 (+16383 * +16383)
02050    2050      37777 1                DS       %37777         ; check #01 (+16383 * -16383)
02051    2051      40000 0                DS       %40000         ; check #02 (-16383 * +16383)
02052    2052      40000 0                DS       %40000         ; check #03 (-16383 * -16383)
02053    2053      00000 1                DS       %00000         ; check #04 (+0 * +0)
02054    2054      00000 1                DS       %00000         ; check #05 (+0 * -0)
02055    2055      77777 0                DS       %77777         ; check #06 (-0 * +0)
02056    2056      77777 0                DS       %77777         ; check #07 (-0 * -0)
                          ; randomly selected checks (one word product)
02057    2057      00007 0                DS       %00007         ; check #08 (7 * 17)
02060    2060      00021 1                DS       %00021         ; check #09 (17 * 7)
02061    2061      00035 1                DS       %00035         ; check #10 (29 * 41)
02062    2062      00051 0                DS       %00051         ; check #11 (41 * 29)
02063    2063      00065 1                DS       %00065         ; check #12 (53 * 67)
02064    2064      00103 0                DS       %00103         ; check #13 (67 * 53)
02065    2065      00117 0                DS       %00117         ; check #14 (79 * 97)
02066    2066      00141 0                DS       %00141         ; check #15 (97 * 79)
02067    2067      00153 0                DS       %00153         ; check #16 (107 * 127)
02070    2070      00177 0                DS       %00177         ; check #17 (127 * 107)
                          ; randomly selected checks (two word product)
02071    2071      00375 0                DS       %00375         ; check #18 (253 * 197)
02072    2072      00305 1                DS       %00305         ; check #19 (197 * 253)
02073    2073      00655 1                DS       %00655         ; check #20 (429 * 351)
02074    2074      00537 0                DS       %00537         ; check #21 (351 * 429)
02075    2075      02455 1                DS       %02455         ; check #22 (1325 * 1067)
02076    2076      02053 0                DS       %02053         ; check #23 (1067 * 1325)
02077    2077      11151 1                DS       %11151         ; check #24 (4713 * 3605)
02100    2100      07025 1                DS       %07025         ; check #25 (3605 * 4713)
02101    2101      20032 1                DS       %20032         ; check #26 (8218 * 7733)
02102    2102      17065 1                DS       %17065         ; check #27 (7733 * 8218)
02103    2103      30273 1                DS       %30273         ; check #28 (12475 * 11501)
02104    2104      26355 0                DS       %26355         ; check #29 (11501 * 12475)
02105    2105      37553 0                DS       %37553         ; check #30 (16235 * 15372)
02106    2106      36014 1                DS       %36014         ; check #31 (15372 * 16235)

                          ; C(K) test values
                          mp2            EQU      *
                          ; check boundary conditions
02107    2107      37777 1                DS       %37777         ; check #00 (+16383 * +16383)
```

```
02110    2110     40000 0                DS       %40000         ; check #01 (+16383 * -16383)
02111    2111     37777 1                DS       %37777         ; check #02 (-16383 * +16383)
02112    2112     40000 0                DS       %40000         ; check #03 (-16383 * -16383)
02113    2113     00000 1                DS       %00000         ; check #04 (+0 * +0)
02114    2114     77777 0                DS       %77777         ; check #05 (+0 * -0)
02115    2115     00000 1                DS       %00000         ; check #06 (-0 * +0)
02116    2116     77777 0                DS       %77777         ; check #07 (-0 * -0)
                          ; randomly selected checks (one word product)
02117    2117     00021 1                DS       %00021         ; check #08 (7 * 17)
02120    2120     00007 0                DS       %00007         ; check #09 (17 * 7)
02121    2121     00051 0                DS       %00051         ; check #10 (29 * 41)
02122    2122     00035 1                DS       %00035         ; check #11 (41 * 29)
02123    2123     00103 0                DS       %00103         ; check #12 (53 * 67)
02124    2124     00065 1                DS       %00065         ; check #13 (67 * 53)
02125    2125     00141 0                DS       %00141         ; check #14 (79 * 97)
02126    2126     00117 0                DS       %00117         ; check #15 (97 * 79)
02127    2127     00177 0                DS       %00177         ; check #16 (107 * 127)
02130    2130     00153 0                DS       %00153         ; check #17 (127 * 107)
                          ; randomly selected checks (two word product)
02131    2131     00305 1                DS       %00305         ; check #18 (253 * 197)
02132    2132     00375 0                DS       %00375         ; check #19 (197 * 253)
02133    2133     00537 0                DS       %00537         ; check #20 (429 * 351)
02134    2134     00655 1                DS       %00655         ; check #21 (351 * 429)
02135    2135     02053 0                DS       %02053         ; check #22 (1325 * 1067)
02136    2136     02455 1                DS       %02455         ; check #23 (1067 * 1325)
02137    2137     07025 1                DS       %07025         ; check #24 (4713 * 3605)
02140    2140     11151 1                DS       %11151         ; check #25 (3605 * 4713)
02141    2141     17065 1                DS       %17065         ; check #26 (8218 * 7733)
02142    2142     20032 1                DS       %20032         ; check #27 (7733 * 8218)
02143    2143     26355 0                DS       %26355         ; check #28 (12475 * 11501)
02144    2144     30273 1                DS       %30273         ; check #29 (11501 * 12475)
02145    2145     36014 1                DS       %36014         ; check #30 (16235 * 15372)
02146    2146     37553 0                DS       %37553         ; check #31 (15372 * 16235)

                          ; A = upper product
                          MPchkA         EQU      *
                          ; check boundary conditions
02147    2147     37776 0                DS       %37776         ; check #00
02150    2150     40001 1                DS       %40001         ; check #01
02151    2151     40001 1                DS       %40001         ; check #02
02152    2152     37776 0                DS       %37776         ; check #03
02153    2153     00000 1                DS       %00000         ; check #04
02154    2154     77777 0                DS       %77777         ; check #05
02155    2155     77777 0                DS       %77777         ; check #06
02156    2156     00000 1                DS       %00000         ; check #07
                          ; randomly selected checks
02157    2157     00000 1                DS       %00000         ; check #08 (7 * 17)
02160    2160     00000 1                DS       %00000         ; check #09 (17 * 7)
02161    2161     00000 1                DS       %00000         ; check #10 (29 * 41)
02162    2162     00000 1                DS       %00000         ; check #11 (41 * 29)
02163    2163     00000 1                DS       %00000         ; check #12 (53 * 67)
02164    2164     00000 1                DS       %00000         ; check #13 (67 * 53)
02165    2165     00000 1                DS       %00000         ; check #14 (79 * 97)
02166    2166     00000 1                DS       %00000         ; check #15 (97 * 79)
02167    2167     00000 1                DS       %00000         ; check #16 (107 * 127)
02170    2170     00000 1                DS       %00000         ; check #17 (127 * 107)
                          ; randomly selected checks (two word product)
02171    2171     00003 1                DS       %00003         ; check #18 (253 * 197)
02172    2172     00003 1                DS       %00003         ; check #19 (197 * 253)
02173    2173     00011 1                DS       %00011         ; check #20 (429 * 351)
02174    2174     00011 1                DS       %00011         ; check #21 (351 * 429)
02175    2175     00126 1                DS       %00126         ; check #22 (1325 * 1067)
02176    2176     00126 1                DS       %00126         ; check #23 (1067 * 1325)
02177    2177     02015 1                DS       %02015         ; check #24 (4713 * 3605)
02200    2200     02015 1                DS       %02015         ; check #25 (3605 * 4713)
02201    2201     07446 0                DS       %07446         ; check #26 (8218 * 7733)
02202    2202     07446 0                DS       %07446         ; check #27 (7733 * 8218)
02203    2203     21065 1                DS       %21065         ; check #28 (12475 * 11501)
02204    2204     21065 1                DS       %21065         ; check #29 (11501 * 12475)
02205    2205     35600 1                DS       %35600         ; check #30 (16235 * 15372)
02206    2206     35600 1                DS       %35600         ; check #31 (15372 * 16235)

                          ; LP = lower product
                          MPchkLP        EQU      *
                          ; check boundary conditions
02207    2207     00001 0                DS       %00001         ; check #00
```

```
02210    2210     77776 1                DS       %77776         ; check #01
02211    2211     77776 1                DS       %77776         ; check #02
02212    2212     00001 0                DS       %00001         ; check #03
02213    2213     00000 1                DS       %00000         ; check #04
02214    2214     77777 0                DS       %77777         ; check #05
02215    2215     77777 0                DS       %77777         ; check #06
02216    2216     00000 1                DS       %00000         ; check #07
                          ; randomly selected checks
02217    2217     00167 1                DS       %00167         ; check #08 (7 * 17)
02220    2220     00167 1                DS       %00167         ; check #09 (17 * 7)
02221    2221     02245 0                DS       %02245         ; check #10 (29 * 41)
02222    2222     02245 0                DS       %02245         ; check #11 (41 * 29)
02223    2223     06737 1                DS       %06737         ; check #12 (53 * 67)
02224    2224     06737 1                DS       %06737         ; check #13 (67 * 53)
02225    2225     16757 0                DS       %16757         ; check #14 (79 * 97)
02226    2226     16757 0                DS       %16757         ; check #15 (97 * 79)
02227    2227     32425 0                DS       %32425         ; check #16 (107 * 127)
02230    2230     32425 0                DS       %32425         ; check #17 (127 * 107)
                          ; randomly selected checks (two word product)
02231    2231     01261 0                DS       %01261         ; check #18 (253 * 197)
02232    2232     01261 0                DS       %01261         ; check #19 (197 * 253)
02233    2233     06063 1                DS       %06063         ; check #20 (429 * 351)
02234    2234     06063 1                DS       %06063         ; check #21 (351 * 429)
02235    2235     11217 0                DS       %11217         ; check #22 (1325 * 1067)
02236    2236     11217 0                DS       %11217         ; check #23 (1067 * 1325)
02237    2237     00235 0                DS       %00235         ; check #24 (4713 * 3605)
02240    2240     00235 0                DS       %00235         ; check #24 (3605 * 4713)
02241    2241     30542 1                DS       %30542         ; check #26 (8218 * 7733)
02242    2242     30542 1                DS       %30542         ; check #27 (7733 * 8218)
02243    2243     00437 1                DS       %00437         ; check #28 (12475 * 11501)
02244    2244     00437 1                DS       %00437         ; check #29 (11501 * 12475)
02245    2245     06404 1                DS       %06404         ; check #30 (16235 * 15372)
02246    2246     06404 1                DS       %06404         ; check #31 (15372 * 16235)

                          chkMP          EQU      *
02247    2247 3  0,0001 0                XCH      Q
02250    2250 5  0,0101 1                TS       savQ           ; save return address

02251    2251 3  1,2045 1                CAF      MPcode
02252    2252 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; Decrementing loop
                          ;   - always executes at least once (tests at end of loop)

                          ;   - loops 'MPstart+1' times; decrements MPindex
02253    2253 3  1,2046 1                XCH      MPstart        ; initialize loop counter

                          ;-----------------------------
                          ; MP check starts here
                          ; uses MPindex to access test values
                          MPloop         EQU      *
02254    2254 5  0,0102 1                TS       MPindex        ; save new index

02255    2255 3  2,5777 0                CAF      EXTENDER
02256    2256 6  0,0102 1                AD       MPindex
02257    2257 5  0,0103 0                TS       MPXTND

02260    2260 2  0,0102 0                INDEX    MPindex
02261    2261 3  1,2047 0                CAF      mp1
02262    2262 2  0,0103 1                INDEX    MPXTND         ; EXTEND using MPindex
02263    2263 4  1,2107 1                MP       mp2

                          ; verify C(A)
02264    2264 4  0,0000 0                COM                     ; get -A
02265    2265 2  0,0102 0                INDEX    MPindex
02266    2266 6  1,2147 1                AD       MPchkA         ; put (-A) + expected value in A
02267    2267 1  0,0000 0                CCS      A              ; compare
02270    2270 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02271    2271 0  1,2036 0                TC       fail           ; +0
02272    2272 0  1,2036 0                TC       fail           ; <0 (A > expected value)

                          ; verify C(LP)
02273    2273 4  0,0003 0                CS       LP             ; get -A
02274    2274 2  0,0102 0                INDEX    MPindex
02275    2275 6  1,2207 0                AD       MPchkLP        ; put (-A) + expected value in A
02276    2276 1  0,0000 0                CCS      A              ; compare
```

```
02277    2277 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02300    2300 0  1,2036 0                TC       fail           ; +0
02301    2301 0  1,2036 0                TC       fail           ; <0 (A > expected value)

                          ; end of MP check
                          ;-----------------------------

02302    2302 1  0,0102 0                CCS      MPindex        ; done?
02303    2303 0  1,2254 0                TC       MPloop         ; not yet, do next check

02304    2304 3  0,0101 1                XCH      savQ
02305    2305 5  0,0001 0                TS       Q              ; restore return address
02306    2306 0  0,0000 0                RETURN
                          ; ----------------------------------------------
                          ; TEST DV INSTRUCTION SUBROUTINE
                          ; L:       DV     K
                          ; Verifies the following:
                          ; - Set C(A) = b(A) / C(K)
                          ; - Set C(Q) = - abs(remainder)
                          ; - Set C(LP) > 0 if quotient is positive
                          ; - Set C(LP) < 0 if quotient is negative
                          ; - Take next instruction from L+1

02307    2307     00002 0 DVcode         DS       DVtst          ; code for this test

                          ; DV test values
                          ;
02310    2310     00037 0 DVstart        DS       31             ; loop DVstart+1 times

                          ; C(A) test values
                          div1           EQU      *
02311    2311     00000 1                DS       %00000         ; check #00 (+0/+0)
02312    2312     00000 1                DS       %00000         ; check #01 (+0/-0)
02313    2313     77777 0                DS       %77777         ; check #02 (-0/+0)
02314    2314     77777 0                DS       %77777         ; check #03 (-0/-0)

02315    2315     00000 1                DS       %00000         ; check #04 (+0/+1)
02316    2316     00000 1                DS       %00000         ; check #05 (+0/-1)
02317    2317     77777 0                DS       %77777         ; check #06 (-0/+1)
02320    2320     77777 0                DS       %77777         ; check #07 (-0/-1)

02321    2321     00000 1                DS       %00000         ; check #08 (+0/+16383)
02322    2322     00000 1                DS       %00000         ; check #09 (+0/-16383)
02323    2323     77777 0                DS       %77777         ; check #10 (-0/+16383)
02324    2324     77777 0                DS       %77777         ; check #11 (-0/-16383)

02325    2325     37776 0                DS       %37776         ; check #12 (+16382/+16383)
02326    2326     37776 0                DS       %37776         ; check #13 (+16382/-16383)
02327    2327     40001 1                DS       %40001         ; check #14 (-16382/+16383)
02330    2330     40001 1                DS       %40001         ; check #15 (-16382/-16383)

02331    2331     37777 1                DS       %37777         ; check #16 (+16383/+16383)
02332    2332     37777 1                DS       %37777         ; check #17 (+16383/-16383)
02333    2333     40000 0                DS       %40000         ; check #18 (-16383/+16383)
02334    2334     40000 0                DS       %40000         ; check #19 (-16383/-16383)

02335    2335     00001 0                DS       %00001         ; check #20 (+1/+2)
02336    2336     00001 0                DS       %00001         ; check #21 (+1/+3)
02337    2337     00001 0                DS       %00001         ; check #22 (+1/+4)
02340    2340     00001 0                DS       %00001         ; check #23 (+1/+5)
02341    2341     00001 0                DS       %00001         ; check #24 (+1/+6)
02342    2342     00001 0                DS       %00001         ; check #25 (+1/+7)
02343    2343     00001 0                DS       %00001         ; check #26 (+1/+8)

02344    2344     00001 0                DS       %00001         ; check #27 (+1/+6)
02345    2345     00002 0                DS       %00002         ; check #28 (+2/+12)
02346    2346     00004 0                DS       %00004         ; check #29 (+4/+24)
02347    2347     00010 0                DS       %00010         ; check #30 (+8/+48)
02350    2350     00020 0                DS       %00020         ; check #31 (+16/+96)

                          ; C(K) test values
                          div2           EQU      *
02351    2351     00000 1                DS       %00000         ; check #00 (+0/+0)
02352    2352     77777 0                DS       %77777         ; check #01 (+0/-0)
02353    2353     00000 1                DS       %00000         ; check #02 (-0/+0)
02354    2354     77777 0                DS       %77777         ; check #03 (-0/-0)
```

```
02355    2355     00001 0                DS       %00001         ; check #04 (+0/+1)
02356    2356     77776 1                DS       %77776         ; check #05 (+0/-1)
02357    2357     00001 0                DS       %00001         ; check #06 (-0/+1)
02360    2360     77776 1                DS       %77776         ; check #07 (-0/-1)

02361    2361     37777 1                DS       %37777         ; check #08 (+0/+16383)
02362    2362     40000 0                DS       %40000         ; check #09 (+0/-16383)
02363    2363     37777 1                DS       %37777         ; check #10 (-0/+16383)
02364    2364     40000 0                DS       %40000         ; check #11 (-0/-16383)

02365    2365     37777 1                DS       %37777         ; check #12 (+16382/+16383)
02366    2366     40000 0                DS       %40000         ; check #13 (+16382/-16383)
02367    2367     37777 1                DS       %37777         ; check #14 (-16382/+16383)
02370    2370     40000 0                DS       %40000         ; check #15 (-16382/-16383)

02371    2371     37777 1                DS       %37777         ; check #16 (+16383/+16383)
02372    2372     40000 0                DS       %40000         ; check #17 (+16383/-16383)
02373    2373     37777 1                DS       %37777         ; check #18 (-16383/+16383)
02374    2374     40000 0                DS       %40000         ; check #19 (-16383/-16383)

02375    2375     00002 0                DS       %00002         ; check #20 (+1/+2)
02376    2376     00003 1                DS       %00003         ; check #21 (+1/+3)
02377    2377     00004 0                DS       %00004         ; check #22 (+1/+4)
02400    2400     00005 1                DS       %00005         ; check #23 (+1/+5)
02401    2401     00006 1                DS       %00006         ; check #24 (+1/+6)
02402    2402     00007 0                DS       %00007         ; check #25 (+1/+7)
02403    2403     00010 0                DS       %00010         ; check #26 (+1/+8)

02404    2404     00006 1                DS       %00006         ; check #27 (+1/+6)
02405    2405     00014 1                DS       %00014         ; check #28 (+2/+12)
02406    2406     00030 1                DS       %00030         ; check #29 (+4/+24)
02407    2407     00060 1                DS       %00060         ; check #30 (+8/+48)
02410    2410     00140 1                DS       %00140         ; check #31 (+16/+96)

                          ; A = quotient
                          DVchkA         EQU      *
02411    2411     37777 1                DS       %37777         ; check #00 (+0/+0)
02412    2412     40000 0                DS       %40000         ; check #01 (+0/-0)
02413    2413     40000 0                DS       %40000         ; check #02 (-0/+0)
02414    2414     37777 1                DS       %37777         ; check #03 (-0/-0)

02415    2415     00000 1                DS       %00000         ; check #04 (+0/+1)
02416    2416     77777 0                DS       %77777         ; check #05 (+0/-1)
02417    2417     77777 0                DS       %77777         ; check #06 (-0/+1)
02420    2420     00000 1                DS       %00000         ; check #07 (-0/-1)

02421    2421     00000 1                DS       %00000         ; check #08 (+0/+16383)
02422    2422     77777 0                DS       %77777         ; check #09 (+0/-16383)
02423    2423     77777 0                DS       %77777         ; check #10 (-0/+16383)
02424    2424     00000 1                DS       %00000         ; check #11 (-0/-16383)

02425    2425     37776 0                DS       %37776         ; check #12 (+16382/+16383)
02426    2426     40001 1                DS       %40001         ; check #13 (+16382/-16383)
02427    2427     40001 1                DS       %40001         ; check #14 (-16382/+16383)
02430    2430     37776 0                DS       %37776         ; check #15 (-16382/-16383)

02431    2431     37777 1                DS       %37777         ; check #16 (+16383/+16383)
02432    2432     40000 0                DS       %40000         ; check #17 (+16383/-16383)
02433    2433     40000 0                DS       %40000         ; check #18 (-16383/+16383)
02434    2434     37777 1                DS       %37777         ; check #19 (-16383/-16383)

02435    2435     20000 0                DS       %20000         ; check #20 (+1/+2)
02436    2436     12525 0                DS       %12525         ; check #21 (+1/+3)
02437    2437     10000 0                DS       %10000         ; check #22 (+1/+4)
02440    2440     06314 1                DS       %06314         ; check #23 (+1/+5)
02441    2441     05252 1                DS       %05252         ; check #24 (+1/+6)
02442    2442     04444 1                DS       %04444         ; check #25 (+1/+7)
02443    2443     04000 0                DS       %04000         ; check #26 (+1/+8)

02444    2444     05252 1                DS       %05252         ; check #27 (+1/+6)
02445    2445     05252 1                DS       %05252         ; check #28 (+2/+12)
02446    2446     05252 1                DS       %05252         ; check #29 (+4/+24)
02447    2447     05252 1                DS       %05252         ; check #30 (+8/+48)
02450    2450     05252 1                DS       %05252         ; check #31 (+16/+96)
```

```
                          ; Q = remainder
                          DVchkQ         EQU      *
02451    2451     77777 0                DS       %77777         ; check #00 (+0/+0)
02452    2452     77777 0                DS       %77777         ; check #01 (+0/-0)
02453    2453     77777 0                DS       %77777         ; check #02 (-0/+0)
02454    2454     77777 0                DS       %77777         ; check #03 (-0/-0)

02455    2455     77777 0                DS       %77777         ; check #04 (+0/+1)
02456    2456     77777 0                DS       %77777         ; check #05 (+0/-1)
02457    2457     77777 0                DS       %77777         ; check #06 (-0/+1)
02460    2460     77777 0                DS       %77777         ; check #07 (-0/-1)

02461    2461     77777 0                DS       %77777         ; check #08 (+0/+16383)
02462    2462     77777 0                DS       %77777         ; check #09 (+0/-16383)
02463    2463     77777 0                DS       %77777         ; check #10 (-0/+16383)
02464    2464     77777 0                DS       %77777         ; check #11 (-0/-16383)

02465    2465     40001 1                DS       %40001         ; check #12 (+16382/+16383)
02466    2466     40001 1                DS       %40001         ; check #13 (+16382/-16383)
02467    2467     40001 1                DS       %40001         ; check #14 (-16382/+16383)
02470    2470     40001 1                DS       %40001         ; check #15 (-16382/-16383)

02471    2471     40000 0                DS       %40000         ; check #16 (+16383/+16383)
02472    2472     40000 0                DS       %40000         ; check #17 (+16383/-16383)
02473    2473     40000 0                DS       %40000         ; check #18 (-16383/+16383)
02474    2474     40000 0                DS       %40000         ; check #19 (-16383/-16383)

02475    2475     77777 0                DS       %77777         ; check #20 (+1/+2)
02476    2476     77776 1                DS       %77776         ; check #21 (+1/+3)
02477    2477     77777 0                DS       %77777         ; check #22 (+1/+4)
02500    2500     77773 1                DS       %77773         ; check #23 (+1/+5)
02501    2501     77773 1                DS       %77773         ; check #24 (+1/+6)
02502    2502     77773 1                DS       %77773         ; check #25 (+1/+7)
02503    2503     77777 0                DS       %77777         ; check #26 (+1/+8)

02504    2504     77773 1                DS       %77773         ; check #27 (+1/+6)
02505    2505     77767 1                DS       %77767         ; check #28 (+2/+12)
02506    2506     77757 1                DS       %77757         ; check #29 (+4/+24)
02507    2507     77737 1                DS       %77737         ; check #30 (+8/+48)
02510    2510     77677 1                DS       %77677         ; check #31 (+16/+96)

                          ; LP = sign
                          DVchkLP        EQU      *
02511    2511     00001 0                DS       %00001         ; check #00 (+0/+0)
02512    2512     40000 0                DS       %40000         ; check #01 (+0/-0)
02513    2513     40001 1                DS       %40001         ; check #02 (-0/+0)
02514    2514     00001 0                DS       %00001         ; check #03 (-0/-0)

02515    2515     00001 0                DS       %00001         ; check #04 (+0/+1)
02516    2516     40000 0                DS       %40000         ; check #05 (+0/-1)
02517    2517     40001 1                DS       %40001         ; check #06 (-0/+1)
02520    2520     00001 0                DS       %00001         ; check #07 (-0/-1)

02521    2521     00001 0                DS       %00001         ; check #08 (+0/+16383)
02522    2522     40000 0                DS       %40000         ; check #09 (+0/-16383)
02523    2523     40001 1                DS       %40001         ; check #10 (-0/+16383)
02524    2524     00001 0                DS       %00001         ; check #11 (-0/-16383)

02525    2525     00001 0                DS       %00001         ; check #12 (+16382/+16383)
02526    2526     40000 0                DS       %40000         ; check #13 (+16382/-16383)
02527    2527     40001 1                DS       %40001         ; check #14 (-16382/+16383)
02530    2530     00001 0                DS       %00001         ; check #15 (-16382/-16383)

02531    2531     00001 0                DS       %00001         ; check #16 (+16383/+16383)
02532    2532     40000 0                DS       %40000         ; check #17 (+16383/-16383)
02533    2533     40001 1                DS       %40001         ; check #18 (-16383/+16383)
02534    2534     00001 0                DS       %00001         ; check #19 (-16383/-16383)

02535    2535     00001 0                DS       %00001         ; check #20 (+1/+2)
02536    2536     00001 0                DS       %00001         ; check #21 (+1/+3)
02537    2537     00001 0                DS       %00001         ; check #22 (+1/+4)
02540    2540     00001 0                DS       %00001         ; check #23 (+1/+5)
02541    2541     00001 0                DS       %00001         ; check #24 (+1/+6)
02542    2542     00001 0                DS       %00001         ; check #25 (+1/+7)
02543    2543     00001 0                DS       %00001         ; check #26 (+1/+8)
```

```
02544    2544     00001 0                DS       %00001         ; check #27 (+1/+6)
02545    2545     00001 0                DS       %00001         ; check #28 (+2/+12)
02546    2546     00001 0                DS       %00001         ; check #29 (+4/+24)
02547    2547     00001 0                DS       %00001         ; check #30 (+8/+48)
02550    2550     00001 0                DS       %00001         ; check #31 (+16/+96)


                          chkDV          EQU      *
02551    2551 3  0,0001 0                XCH      Q
02552    2552 5  0,0101 1                TS       savQ           ; save return address

02553    2553 3  1,2307 1                CAF      DVcode
02554    2554 5  0,0100 0                TS       curtest        ; set code identifying test


                          ; Decrementing loop
                          ;   - always executes at least once (tests at end of loop)

                          ;   - loops 'DVstart+1' times; decrements DVindex
02555    2555 3  1,2310 1                XCH      DVstart        ; initialize loop counter

                          ;-----------------------------

                          ; DV check starts here
                          ; uses DVindex to access test values
                          DVloop         EQU      *
02556    2556 5  0,0105 0                TS       DVindex        ; save new index

02557    2557 3  2,5777 0                CAF      EXTENDER
02560    2560 6  0,0105 0                AD       DVindex
02561    2561 5  0,0106 0                TS       DVXTND

02562    2562 2  0,0105 1                INDEX    DVindex
02563    2563 3  1,2311 0                CAF      div1
02564    2564 2  0,0106 1                INDEX    DVXTND         ; EXTEND using DVindex
02565    2565 5  1,2351 1                DV       div2
02566    2566 5  0,0104 1                TS       DVsavA

                          ; verify C(Q)
02567    2567 4  0,0001 1                CS       Q              ; get -A
02570    2570 2  0,0105 1                INDEX    DVindex
02571    2571 6  1,2451 0                AD       DVchkQ         ; put (-A) + expected value in A
02572    2572 1  0,0000 0                CCS      A              ; compare
02573    2573 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02574    2574 0  1,2036 0                TC       fail           ; +0
02575    2575 0  1,2036 0                TC       fail           ; <0 (A > expected value)

                          ; verify C(A)
02576    2576 4  0,0104 0                CS       DVsavA         ; get -A
02577    2577 2  0,0105 1                INDEX    DVindex
02600    2600 6  1,2411 1                AD       DVchkA         ; put (-A) + expected value in A
02601    2601 1  0,0000 0                CCS      A              ; compare
02602    2602 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02603    2603 0  1,2036 0                TC       fail           ; +0
02604    2604 0  1,2036 0                TC       fail           ; <0 (A > expected value)

                          ; verify C(LP)
02605    2605 4  0,0003 0                CS       LP             ; get -A
02606    2606 2  0,0105 1                INDEX    DVindex
02607    2607 6  1,2511 0                AD       DVchkLP        ; put (-A) + expected value in A
02610    2610 1  0,0000 0                CCS      A              ; compare
02611    2611 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02612    2612 0  1,2036 0                TC       fail           ; +0
02613    2613 0  1,2036 0                TC       fail           ; <0 (A > expected value)

                          ; end of DV check
                          ;-----------------------------

02614    2614 1  0,0105 1                CCS      DVindex        ; done?
02615    2615 0  1,2556 0                TC       DVloop         ; not yet, do next check

02616    2616 3  0,0101 1                XCH      savQ
02617    2617 5  0,0001 0                TS       Q              ; restore return address
02620    2620 0  0,0000 0                RETURN
                          ; ---------------------------------------------
                          ; TEST SU INSTRUCTION SUBROUTINE
                          ; L:        SU      K
                          ; Verifies the following:
                          ; - Set C(A) = b(A) - C(K)
                          ; - Take next instruction from L+1
                          ; - if C(A) has positive overflow,
                          ; -- increment overflow counter by 1
                          ; - if C(A) has negative overflow,
                          ; -- decrement overflow counter by 1

02621    2621     00003 1 SUcode         DS       SUtst          ; code for this test

02622    2622     00000 1 SUplus0        DS       +0
02623    2623     00001 0 SUplus1        DS       1
02624    2624     77776 1 SUmin1         DS       -1

02625    2625     25252 0 SU25252        DS       %25252         ; +10922 decimal
02626    2626     12525 0 SU12525        DS       %12525         ; +5461 decimal
02627    2627     37777 1 SU37777        DS       %37777         ; largest positive number
02630    2630     12524 1 SU12524        DS       %12524         ; + overflow of %25252+%25252

02631    2631     52525 1 SU52525        DS       %52525         ; -10922 decimal
02632    2632     65252 1 SU65252        DS       %65252         ; -5461 decimal
02633    2633     40000 0 SU40000        DS       %40000         ; largest negative number
02634    2634     65253 0 SU65253        DS       %65253         ; - overflow of %52525+65252

                          chkSU          EQU      *
02635    2635 3  0,0001 0                XCH      Q
02636    2636 5  0,0101 1                TS       savQ           ; save return address

02637    2637 3  1,2621 0                CAF      SUcode
02640    2640 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; NOTE: these test are similar to the checks for AD, but
                          ; the AD augend value has been changed to negative and AD has
                          ; been changed to SU. The results produced by this change
                          ; are identical to AD, and so the checks are the same.

                          ; TEST1: difference positive, no overflow
                          ; sub: %25252 - %65252 = %37777 (sign + 14 magnitude)
02641    2641 3  1,2625 1                CAF      SU25252
02642    2642 2  0,0000 1                EXTEND
02643    2643 6  1,2632 1                SU       SU65252
                          ; verify C(A) = %37777
02644    2644 4  0,0000 0                COM                     ; get -A
02645    2645 6  1,2627 0                AD       SU37777        ; put (-A) + expected value in A
02646    2646 1  0,0000 0                CCS      A              ; compare
02647    2647 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02650    2650 0  1,2036 0                TC       fail           ; +0
02651    2651 0  1,2036 0                TC       fail           ; <0 (A > expected value)

                          ; TEST2: difference negative, no overflow (sign + 14 magnitude)
                          ; sub: %52525 - %12525 = %40000
02652    2652 3  1,2631 1                CAF      SU52525
02653    2653 2  0,0000 1                EXTEND
02654    2654 6  1,2626 1                SU       SU12525
                          ; verify C(A) = %40000
02655    2655 4  0,0000 0                COM                     ; get -A
02656    2656 6  1,2633 0                AD       SU40000        ; put (-A) + expected value in A
02657    2657 1  0,0000 0                CCS      A              ; compare
02660    2660 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02661    2661 0  1,2036 0                TC       fail           ; +0
02662    2662 0  1,2036 0                TC       fail           ; <0 (A > expected value)

                          ; TEST3: difference positive, overflow
                          ; initialize overflow counter and positive overflow storage
02663    2663 3  1,2622 0                CAF      SUplus0
02664    2664 5  0,0034 0                TS       OVFCNTR
02665    2665 5  0,0107 1                TS       SUk
                          ; sub: %25252 - %52525 = %52524 (sign + 14 magnitude)
02666    2666 3  1,2625 1                CAF      SU25252
02667    2667 2  0,0000 1                EXTEND
02670    2670 6  1,2631 1                SU       SU52525
02671    2671 5  0,0107 1                TS       SUk            ; store positive overflow
02672    2672 0  1,2036 0                TC       fail
                          ; verify SUk = %12524
02673    2673 4  0,0107 0                CS       SUk            ; get -A
```

```
02674    2674 6  1,2630 0                AD       SU12524        ; put (-A) + expected value in A
02675    2675 1  0,0000 0                CCS      A              ; compare
02676    2676 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02677    2677 0  1,2036 0                TC       fail           ; +0
02700    2700 0  1,2036 0                TC       fail           ; <0 (A > expected value)
                          ; verify overflow counter =%00001
02701    2701 4  0,0034 1                CS       OVFCNTR        ; get -A
02702    2702 6  1,2623 1                AD       SUplus1        ; put (-A) + expected value in A
02703    2703 1  0,0000 0                CCS      A              ; compare
02704    2704 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02705    2705 0  1,2036 0                TC       fail           ; +0
02706    2706 0  1,2036 0                TC       fail           ; <0 (A > expected value)

                          ; TEST4: difference negative, overflow
02707    2707 3  1,2622 0                CAF      SUplus0
02710    2710 5  0,0034 0                TS       OVFCNTR
02711    2711 5  0,0107 1                TS       SUk
                          ; add: %52525 + %25252 = %25253 (sign + 14 magnitude)
02712    2712 3  1,2631 1                CAF      SU52525
02713    2713 2  0,0000 1                EXTEND
02714    2714 6  1,2625 1                SU       SU25252
02715    2715 5  0,0107 1                TS       SUk            ; store negative overflow
02716    2716 0  1,2036 0                TC       fail
                          ; verify SUk = %65253
02717    2717 4  0,0107 0                CS       SUk            ; get -A
02720    2720 6  1,2634 1                AD       SU65253        ; put (-A) + expected value in A
02721    2721 1  0,0000 0                CCS      A              ; compare
02722    2722 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02723    2723 0  1,2036 0                TC       fail           ; +0
02724    2724 0  1,2036 0                TC       fail           ; <0 (A > expected value)
                          ; verify overflow counter =%77776
02725    2725 4  0,0034 1                CS       OVFCNTR        ; get -A
02726    2726 6  1,2624 0                AD       SUmin1         ; put (-A) + expected value in A
02727    2727 1  0,0000 0                CCS      A              ; compare
02730    2730 0  1,2036 0                TC       fail           ; >0 (A < expected value)
02731    2731 0  1,2036 0                TC       fail           ; +0
02732    2732 0  1,2036 0                TC       fail           ; <0 (A > expected value)

02733    2733 3  0,0101 1                XCH      savQ
02734    2734 5  0,0001 0                TS       Q              ; restore return address
02735    2735 0  0,0000 0                RETURN
                          ; ---------------------------------------------
                          ; PASSED ALL TESTS!

02736    2736     12345 0 PASScode       DS       PASS

                          finish         EQU      *
02737    2737 3  1,2736 1                CAF      PASScode
02740    2740 5  0,0100 0                TS       curtest        ; set current test code to PASS
02741    2741 0  0,0000 0                RETURN

                          ; ---------------------------------------------
                          ; INTERRUPT SERVICE ROUTINE

                          goT3           EQU      *
                          goER           EQU      *
                          goDS           EQU      *
                          goKEY          EQU      *
                          goUP           EQU      *


                          endRUPT        EQU      *
02742    2742 3  0,0027 1                XCH      QRUPT          ; restore Q
02743    2743 5  0,0001 0                TS       Q
02744    2744 3  0,0026 0                XCH      ARUPT          ; restore A
02745    2745 2  0,0000 1                RESUME                  ; finished, go back



Assembly complete. Errors = 0

Symbol table:
START          000000   MPtst          000001   DVtst          000002
SUtst          000003   PASS           012345   EXTENDER       005777
OVFCNTR        000034   curtest        000100   savQ           000101
```

```
MPindex        000102   MPXTND         000103   DVsavA         000104
DVindex        000105   DVXTND         000106   SUk            000107
GOPROG         002000   T3RUPT         002004   ERRUPT         002010
DSRUPT         002014   KEYRUPT        002020   UPRUPT         002024
goMAIN         002030   fail           002036   end            002040
STRTcode       002041   begin          002042   MPcode         002045
MPstart        002046   mp1            002047   mp2            002107
MPchkA         002147   MPchkLP        002207   chkMP          002247
MPloop         002254   DVcode         002307   DVstart        002310
div1           002311   div2           002351   DVchkA         002411
DVchkQ         002451   DVchkLP        002511   chkDV          002551
DVloop         002556   SUcode         002621   SUplus0        002622
SUplus1        002623   SUmin1         002624   SU25252        002625
SU12525        002626   SU37777        002627   SU12524        002630
SU52525        002631   SU65252        002632   SU40000        002633
SU65253        002634   chkSU          002635   PASScode       002736
finish         002737   goT3           002742   goER           002742
goDS           002742   goKEY          002742   goUP           002742
endRUPT        002742   ARUPT          000026   Q              000001
QRUPT          000027   A              000000   LP             000003
```

TECO3 assembler listing

```
Block I Apollo Guidance Computer (AGC4) assembler version 1.6 for EPROM

First pass: generate symbol table.
Second pass: generate object code.

                          ; TECO3 (file:teco3.asm)
                          ;
                          ; Version:  1.0
                          ; Author:   John Pultorak
                          ; Date:     9/14/2001
                          ;
                          ; PURPOSE:
                          ; Test and checkout program for the Block 1 Apollo Guidance Computer.
                          ; Tests editing registers: CYR, SR, CYL, SL.
                          ;
                          ; OPERATION:
                          ; Enters an infinite loop at the end of the test. The A register
                          ; contains the code for the test that failed, or the PASS code if all
                          ; tests succeeded. See test codes below.
                          ;
                          ; ERRATA:
                          ; - Written for the AGC4R assembler. The assembler directives and
                          ; syntax differ somewhat from the original AGC assembler.
                          ; - The tests attempt to check all threads, but are not exhaustive.
                          ;
                          ; SOURCES:
                          ; Information on the Block 1 architecture: instruction set, instruction
                          ; sequences, registers, register transfers, control pulses, memory and
                          ; memory addressing, I/O assignments, interrupts, and involuntary
                          ; counters was obtained from:
                          ;
                          ;   A. Hopkins, R. Alonso, and H. Blair-Smith, "Logical Description
                          ;           for the Apollo Guidance Computer (AGC4)", R-393,
                          ;           MIT Instrumentation Laboratory, Cambridge, MA, Mar. 1963.
                          ;
                          ; Supplementary information was obtained from:
                          ;
                          ;   R. Alonso, J. H. Laning, Jr. and H. Blair-Smith, "Preliminary
                          ;           MOD 3C Programmer's Manual", E-1077, MIT Instrumentation
                          ;           Laboratory, Cambridge, MA, Nov. 1961.
                          ;
                          ;   B. I. Savage and A. Drake, "AGC4 Basic Training Manual, Volume I",
                          ;           E-2052, MIT Instrumentation Laboratory, Cambridge,
                          ;           MA, Jan. 1967.
                          ;
                          ;   E. C. Hall, "MIT's Role in Project Apollo, Volume III, Computer
                          ;           Subsystem", R-700, MIT Charles Stark Draper Laboratory,
                          ;           Cambridge, MA, Aug. 1972.
                          ;
                          ;   A. Hopkins, "Guidance Computer Design, Part VI", source unknown.
                          ;
                          ;   A. I. Green and J. J. Rocchio, "Keyboard and Display System Program
                          ;           for AGC (Program Sunrise)", E-1574, MIT Instrumentation
                          ;           Laboratory, Cambridge, MA, Aug. 1964.
                          ;
                          ;   E, C. Hall, "Journey to the Moon: The History of the Apollo
                          ;           Guidance Computer", AIAA, Reston VA, 1996.
                          ;

                          START          EQU      %00

                          CYRtst         EQU      %01            ; CYR check failed
                          SRtst          EQU      %02            ; SR check failed
                          CYLtst         EQU      %03            ; CYL check failed
                          SLtst          EQU      %04            ; SL check failed

                          PASS           EQU      %12345         ; PASSED all checks
                          ; ----------------------------------------------

                                         ORG      EXTENDER
05777    5777     47777 0                DS       %47777         ; needed for EXTEND
```

```
                          ; ----------------------------------------------
                          ; ERASEABLE MEMORY -- DATA SEGMENT

                                         ORG      %100           ; start of data area
00100    0100     00000 1 curtest        DS       START          ; current test
00101    0101     00000 1 savQ           DS       %0

                          ; CYR test values
00102    0102     00000 1 CYRval         DS       %0             ; current test value
00103    0103     00000 1 iCYR           DS       %0             ; current index

                          ; SR test values
00104    0104     00000 1 SRval          DS       %0             ; current test value
00105    0105     00000 1 iSR            DS       %0             ; current index

                          ; CYL test values
00106    0106     00000 1 CYLval         DS       %0             ; current test value
00107    0107     00000 1 iCYL           DS       %0             ; current index

                          ; SL test values
00110    0110     00000 1 SLval          DS       %0             ; current test value
00111    0111     00000 1 iSL            DS       %0             ; current index

                          ; ----------------------------------------------
                          ; ENTRY POINTS

                          ; program (re)start
                                         ORG      GOPROG
02000    2000 0  1,2030 0                TC       goMAIN

                          ; interrupt service entry points
                                         ORG      T3RUPT
02004    2004 5  0,0026 0                TS       ARUPT
02005    2005 3  0,0001 0                XCH      Q
02006    2006 5  0,0027 1                TS       QRUPT
02007    2007 0  1,2424 1                TC       goT3

                                         ORG      ERRUPT
02010    2010 5  0,0026 0                TS       ARUPT
02011    2011 3  0,0001 0                XCH      Q
02012    2012 5  0,0027 1                TS       QRUPT
02013    2013 0  1,2424 1                TC       goER

                                         ORG      DSRUPT
02014    2014 5  0,0026 0                TS       ARUPT
02015    2015 3  0,0001 0                XCH      Q
02016    2016 5  0,0027 1                TS       QRUPT
02017    2017 0  1,2424 1                TC       goDS

                                         ORG      KEYRUPT
02020    2020 5  0,0026 0                TS       ARUPT
02021    2021 3  0,0001 0                XCH      Q
02022    2022 5  0,0027 1                TS       QRUPT
02023    2023 0  1,2424 1                TC       goKEY


                                         ORG      UPRUPT
02024    2024 5  0,0026 0                TS       ARUPT
02025    2025 3  0,0001 0                XCH      Q
02026    2026 5  0,0027 1                TS       QRUPT
02027    2027 0  1,2424 1                TC       goUP


                          ; ----------------------------------------------
                          ; FIXED MEMORY -- SHARED DATA SEGMENT

                          ; ----------------------------------------------
                          ; MAIN PROGRAM

                          goMAIN         EQU      *
02030    2030 2  0,0000 0                INHINT                  ; disable interrupts

02031    2031 0  1,2043 1                TCR      begin

                          ; Test extracode instructions.
02032    2032 0  1,2070 1                TCR      chkCYR
```

```
02033    2033 0  1,2162 0               TCR      chkSR
02034    2034 0  1,2255 1               TCR      chkCYL
02035    2035 0  1,2347 0               TCR      chkSL

                         ; Passed all tests.
02036    2036 0  1,2421 1               TCR      finish

                         fail           EQU      *
02037    2037 3  0,0100 0               XCH      curtest        ; load last passed test into A
02040    2040 5  0,0100 0               TS       curtest

                         end            EQU      *
02041    2041 0  1,2041 0               TC       end            ; finished, TC trap

                         ; -----------------------------------------------
                         ; INITIALIZE FOR START OF TESTING

02042    2042     00000 1 STRTcode       DS       START

                         begin          EQU      *
02043    2043 3  1,2042 0               XCH      STRTcode
02044    2044 5  0,0100 0               TS       curtest        ; set current test code to START
02045    2045 0  0,0000 0               RETURN

                         ; -----------------------------------------------
                         ; TEST CYR EDITING FUNCTION SUBROUTINE
                         ; Rotate a test value right through CYR 15 times.
                         ; Test the value against an expected value for each time.
                         ; After 15 rotations, the value should equal the initial
                         ; value.

02046    2046     00001 0 CYRcode        DS       CYRtst         ; code for this test

                         ; CYR test values
02047    2047     03431 1 CYRinit        DS       %03431         ; init test value
02050    2050     00016 0 CYRindx        DS       14             ; loop CYRindx+1 times

                         ; check CYR against these values
                         CYRbase        EQU      *
02051    2051     03431 1               DS       %03431         ; check #0 (back to start)
02052    2052     07062 1               DS       %07062         ; check #1
02053    2053     16144 1               DS       %16144         ; check #2
02054    2054     34310 1               DS       %34310         ; check #3
02055    2055     70620 1               DS       %70620         ; check #4
02056    2056     61441 1               DS       %61441         ; check #5
02057    2057     43103 1               DS       %43103         ; check #6
02060    2060     06207 1               DS       %06207         ; check #7
02061    2061     14416 1               DS       %14416         ; check #8
02062    2062     31034 1               DS       %31034         ; check #9
02063    2063     62070 1               DS       %62070         ; check #10
02064    2064     44161 1               DS       %44161         ; check #11
02065    2065     10343 1               DS       %10343         ; check #12
02066    2066     20706 1               DS       %20706         ; check #13
02067    2067     41614 1               DS       %41614         ; check #14

                         chkCYR         EQU      *
02070    2070 3  0,0001 0               XCH      Q
02071    2071 5  0,0101 1               TS       savQ           ; save return address

02072    2072 3  1,2046 1               CAF      CYRcode
02073    2073 5  0,0100 0               TS       curtest        ; set test code to this test

02074    2074 3  1,2047 0               XCH      CYRinit        ; init value to rotate
02075    2075 5  0,0102 1               TS       CYRval

02076    2076 3  1,2050 0               XCH      CYRindx        ; load init index

                         CYRloop        EQU      *
02077    2077 5  0,0103 0               TS       iCYR           ; save index

                         ; rotate A right (CYR)
02100    2100 3  0,0102 1               XCH      CYRval
02101    2101 5  0,0020 0               TS       CYR            ; rotate
02102    2102 3  0,0020 0               XCH      CYR            ; put result in A
02103    2103 5  0,0102 1               TS       CYRval
```

```
                          ; verify C(A)
02104    2104 4  0,0000 0                COM                     ; get -A
02105    2105 2  0,0103 1                INDEX    iCYR
02106    2106 6  1,2051 1                AD       CYRbase        ; put (-A) + expected value in A
02107    2107 1  0,0000 0                CCS      A              ; compare
02110    2110 0  1,2037 1                TC       fail           ; >0 (A < expected value)
02111    2111 0  1,2037 1                TC       fail           ; +0
02112    2112 0  1,2037 1                TC       fail           ; <0 (A > expected value)

                          ; loop back to test next value
02113    2113 1  0,0103 1                CCS      iCYR           ; done?
02114    2114 0  1,2077 0                TC       CYRloop        ; not yet, do next check

02115    2115 3  0,0101 1                XCH      savQ
02116    2116 5  0,0001 0                TS       Q              ; restore return address
02117    2117 0  0,0000 0                RETURN

                          ; ---------------------------------------------
                          ; TEST SR EDITING FUNCTION SUBROUTINE
                          ; Shift a test value right through SR 15 times.
                          ; Test the value against an expected value for each time.
                          ; After 15 shifts, the value should equal the sign (SG).

02120    2120     00002 0 SRcode         DS       SRtst          ; code for this test

                          ; SR test values
02121    2121     03431 1 SRinitP        DS       %03431         ; positive init test value
02122    2122     44346 0 SRinitN        DS       %44346         ; negative init test value
02123    2123     00016 0 SRindx         DS       14             ; loop SRindx+1 times

                          ; check SR against these values (positive)
                          SRbaseP        EQU      *
02124    2124     00000 1                DS       %00000         ; check #0 (back to start)
02125    2125     00000 1                DS       %00000         ; check #1
02126    2126     00000 1                DS       %00000         ; check #2
02127    2127     00000 1                DS       %00000         ; check #3
02130    2130     00000 1                DS       %00000         ; check #4
02131    2131     00001 0                DS       %00001         ; check #5
02132    2132     00003 1                DS       %00003         ; check #6
02133    2133     00007 0                DS       %00007         ; check #7
02134    2134     00016 0                DS       %00016         ; check #8
02135    2135     00034 0                DS       %00034         ; check #9
02136    2136     00070 0                DS       %00070         ; check #10
02137    2137     00161 1                DS       %00161         ; check #11
02140    2140     00343 0                DS       %00343         ; check #12
02141    2141     00706 0                DS       %00706         ; check #13
02142    2142     01614 0                DS       %01614         ; check #14

                          ; check SR against these values (negative)
                          SRbaseN        EQU      *
02143    2143     77777 0                DS       %77777         ; check #0 (back to start)
02144    2144     77777 0                DS       %77777         ; check #1
02145    2145     77776 1                DS       %77776         ; check #2
02146    2146     77774 0                DS       %77774         ; check #3
02147    2147     77771 0                DS       %77771         ; check #4
02150    2150     77762 1                DS       %77762         ; check #5
02151    2151     77744 0                DS       %77744         ; check #6
02152    2152     77710 1                DS       %77710         ; check #7
02153    2153     77621 1                DS       %77621         ; check #8
02154    2154     77443 1                DS       %77443         ; check #9
02155    2155     77107 1                DS       %77107         ; check #10
02156    2156     76216 0                DS       %76216         ; check #11
02157    2157     74434 1                DS       %74434         ; check #12
02160    2160     71071 1                DS       %71071         ; check #13
02161    2161     62163 1                DS       %62163         ; check #14

                          chkSR          EQU      *
02162    2162 3  0,0001 0                XCH      Q
02163    2163 5  0,0101 1                TS       savQ           ; save return address

02164    2164 3  1,2120 0                CAF      SRcode
02165    2165 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; TEST 1: shift a postive value.
02166    2166 3  1,2121 1                XCH      SRinitP        ; init value to shift
02167    2167 5  0,0104 1                TS       SRval
```

```
02170    2170 3  1,2123 0                XCH      SRindx         ; load init index


                          SRloopP        EQU      *
02171    2171 5  0,0105 0                TS       iSR            ; save index

                          ; shift A right (SR)
02172    2172 3  0,0104 1                XCH      SRval
02173    2173 5  0,0021 1                TS       SR             ; shift
02174    2174 3  0,0021 1                XCH      SR             ; put result in A
02175    2175 5  0,0104 1                TS       SRval

                          ; verify C(A)
02176    2176 4  0,0000 0                COM                     ; get -A
02177    2177 2  0,0105 1                INDEX    iSR
02200    2200 6  1,2124 1                AD       SRbaseP        ; put (-A) + expected value in A
02201    2201 1  0,0000 0                CCS      A              ; compare
02202    2202 0  1,2037 1                TC       fail           ; >0 (A < expected value)
02203    2203 0  1,2037 1                TC       fail           ; +0
02204    2204 0  1,2037 1                TC       fail           ; <0 (A > expected value)

                          ; loop back to test next value
02205    2205 1  0,0105 1                CCS      iSR            ; done?
02206    2206 0  1,2171 1                TC       SRloopP        ; not yet, do next check

                          ; TEST 2: shift a negative value
02207    2207 3  1,2122 1                XCH      SRinitN        ; init value to shift
02210    2210 5  0,0104 1                TS       SRval

02211    2211 3  1,2123 0                XCH      SRindx         ; load init index


                          SRloopN        EQU      *
02212    2212 5  0,0105 0                TS       iSR            ; save index

                          ; shift A left (SR)
02213    2213 3  0,0104 1                XCH      SRval
02214    2214 5  0,0021 1                TS       SR             ; shift
02215    2215 3  0,0021 1                XCH      SR             ; put result in A
02216    2216 5  0,0104 1                TS       SRval

                          ; verify C(A)
02217    2217 4  0,0000 0                COM                     ; get -A
02220    2220 2  0,0105 1                INDEX    iSR
02221    2221 6  1,2143 0                AD       SRbaseN        ; put (-A) + expected value in A
02222    2222 1  0,0000 0                CCS      A              ; compare
02223    2223 0  1,2037 1                TC       fail           ; >0 (A < expected value)
02224    2224 0  1,2037 1                TC       fail           ; +0
02225    2225 0  1,2037 1                TC       fail           ; <0 (A > expected value)

                          ; loop back to test next value
02226    2226 1  0,0105 1                CCS      iSR            ; done?
02227    2227 0  1,2212 1                TC       SRloopN        ; not yet, do next check

02230    2230 3  0,0101 1                XCH      savQ
02231    2231 5  0,0001 0                TS       Q              ; restore return address
02232    2232 0  0,0000 0                RETURN

                          ; -----------------------------------------------
                          ; TEST CYL EDITING FUNCTION SUBROUTINE
                          ; Rotate a test value left through CYL 15 times.
                          ; Test the value against an expected value for each time.
                          ; After 15 rotations, the value should equal the initial
                          ; value.

02233    2233     00003 1 CYLcode        DS       CYLtst         ; code for this test

                          ; CYL test values
02234    2234     03431 1 CYLinit        DS       %03431         ; init test value
02235    2235     00016 0 CYLindx        DS       14             ; loop CYLindx+1 times

                          ; check CYL against these values
                          CYLbase        EQU      *
02236    2236     03431 1                DS       %03431         ; check #0 (back to start)
02237    2237     41614 1                DS       %41614         ; check #1
```

```
02240    2240     20706 1                DS       %20706         ; check #2
02241    2241     10343 1                DS       %10343         ; check #3
02242    2242     44161 1                DS       %44161         ; check #4
02243    2243     62070 1                DS       %62070         ; check #5
02244    2244     31034 1                DS       %31034         ; check #6
02245    2245     14416 1                DS       %14416         ; check #7
02246    2246     06207 1                DS       %06207         ; check #8
02247    2247     43103 1                DS       %43103         ; check #9
02250    2250     61441 1                DS       %61441         ; check #10
02251    2251     70620 1                DS       %70620         ; check #11
02252    2252     34310 1                DS       %34310         ; check #12
02253    2253     16144 1                DS       %16144         ; check #13
02254    2254     07062 1                DS       %07062         ; check #14

                          chkCYL         EQU      *
02255    2255 3  0,0001 0                XCH      Q
02256    2256 5  0,0101 1                TS       savQ           ; save return address

02257    2257 3  1,2233 1                CAF      CYLcode
02260    2260 5  0,0100 0                TS       curtest        ; set test code to this test

02261    2261 3  1,2234 0                XCH      CYLinit        ; init value to rotate
02262    2262 5  0,0106 0                TS       CYLval

02263    2263 3  1,2235 1                XCH      CYLindx        ; load init index

                          CYLloop        EQU      *
02264    2264 5  0,0107 1                TS       iCYL           ; save index

                          ; rotate A left (CYL)
02265    2265 3  0,0106 0                XCH      CYLval
02266    2266 5  0,0022 1                TS       CYL            ; rotate
02267    2267 3  0,0022 1                XCH      CYL            ; put result in A
02270    2270 5  0,0106 0                TS       CYLval

                          ; verify C(A)
02271    2271 4  0,0000 0                COM                     ; get -A
02272    2272 2  0,0107 0                INDEX    iCYL
02273    2273 6  1,2236 1                AD       CYLbase        ; put (-A) + expected value in A
02274    2274 1  0,0000 0                CCS      A              ; compare
02275    2275 0  1,2037 1                TC       fail           ; >0 (A < expected value)
02276    2276 0  1,2037 1                TC       fail           ; +0
02277    2277 0  1,2037 1                TC       fail           ; <0 (A > expected value)

                          ; loop back to test next value
02300    2300 1  0,0107 0                CCS      iCYL           ; done?
02301    2301 0  1,2264 0                TC       CYLloop        ; not yet, do next check

02302    2302 3  0,0101 1                XCH      savQ
02303    2303 5  0,0001 0                TS       Q              ; restore return address
02304    2304 0  0,0000 0                RETURN

                          ; ---------------------------------------------
                          ; TEST SL EDITING FUNCTION SUBROUTINE
                          ; Shift a test value left through SL 15 times.
                          ; Test the value against an expected value for each time.
                          ; After 15 shifts, the value should equal the sign (SG).

02305    2305     00004 0 SLcode         DS       SLtst          ; code for this test

                          ; SL test values
02306    2306     03431 1 SLinitP        DS       %03431         ; positive init test value
02307    2307     44346 0 SLinitN        DS       %44346         ; negative init test value
02310    2310     00016 0 SLindx         DS       14             ; loop SLindx+1 times

                          ; check SL against these values (positive)
                          SLbaseP        EQU      *
02311    2311     00000 1                DS       %00000         ; check #0 (back to start)
02312    2312     00000 1                DS       %00000         ; check #1
02313    2313     20000 0                DS       %20000         ; check #2
02314    2314     10000 0                DS       %10000         ; check #3
02315    2315     04000 0                DS       %04000         ; check #4
02316    2316     22000 1                DS       %22000         ; check #5
02317    2317     31000 0                DS       %31000         ; check #6
02320    2320     14400 0                DS       %14400         ; check #7
02321    2321     06200 0                DS       %06200         ; check #8
```

```
02322    2322     03100 0                DS       %03100         ; check #9
02323    2323     21440 1                DS       %21440         ; check #10
02324    2324     30620 0                DS       %30620         ; check #11
02325    2325     34310 1                DS       %34310         ; check #12
02326    2326     16144 1                DS       %16144         ; check #13
02327    2327     07062 1                DS       %07062         ; check #14

                          ; check SL against these values (negative)
                          SLbaseN        EQU      *
02330    2330     77777 0                DS       %77777         ; check #0 (back to start)
02331    2331     77777 0                DS       %77777         ; check #1
02332    2332     57777 1                DS       %57777         ; check #2
02333    2333     67777 1                DS       %67777         ; check #3
02334    2334     73777 1                DS       %73777         ; check #4
02335    2335     55777 0                DS       %55777         ; check #5
02336    2336     46777 1                DS       %46777         ; check #6
02337    2337     63377 1                DS       %63377         ; check #7
02340    2340     71577 1                DS       %71577         ; check #8
02341    2341     74677 1                DS       %74677         ; check #9
02342    2342     56337 0                DS       %56337         ; check #10
02343    2343     47157 1                DS       %47157         ; check #11
02344    2344     43467 0                DS       %43467         ; check #12
02345    2345     61633 0                DS       %61633         ; check #13
02346    2346     50715 1                DS       %50715         ; check #14

                          chkSL          EQU      *
02347    2347 3  0,0001 0                XCH      Q
02350    2350 5  0,0101 1                TS       savQ           ; save return address

02351    2351 3  1,2305 0                CAF      SLcode
02352    2352 5  0,0100 0                TS       curtest        ; set test code to this test

                          ; TEST 1: shift a postive value.
02353    2353 3  1,2306 0                XCH      SLinitP        ; init value to shift
02354    2354 5  0,0110 1                TS       SLval

02355    2355 3  1,2310 1                XCH      SLindx         ; load init index


                          SLloopP        EQU      *
02356    2356 5  0,0111 0                TS       iSL            ; save index

                          ; shift A left (SL)
02357    2357 3  0,0110 1                XCH      SLval
02360    2360 5  0,0023 0                TS       SL             ; shift
02361    2361 3  0,0023 0                XCH      SL             ; put result in A
02362    2362 5  0,0110 1                TS       SLval

                          ; verify C(A)
02363    2363 4  0,0000 0                COM                     ; get -A
02364    2364 2  0,0111 1                INDEX    iSL
02365    2365 6  1,2311 0                AD       SLbaseP        ; put (-A) + expected value in A
02366    2366 1  0,0000 0                CCS      A              ; compare
02367    2367 0  1,2037 1                TC       fail           ; >0 (A < expected value)
02370    2370 0  1,2037 1                TC       fail           ; +0
02371    2371 0  1,2037 1                TC       fail           ; <0 (A > expected value)

                          ; loop back to test next value
02372    2372 1  0,0111 1                CCS      iSL            ; done?
02373    2373 0  1,2356 0                TC       SLloopP        ; not yet, do next check

                          ; TEST 2: shift a negative value
02374    2374 3  1,2307 1                XCH      SLinitN        ; init value to shift
02375    2375 5  0,0110 1                TS       SLval

02376    2376 3  1,2310 1                XCH      SLindx         ; load init index

                          SLloopN        EQU      *
02377    2377 5  0,0111 0                TS       iSL            ; save index

                          ; shift A left (SL)
02400    2400 3  0,0110 1                XCH      SLval
02401    2401 5  0,0023 0                TS       SL             ; shift
02402    2402 3  0,0023 0                XCH      SL             ; put result in A
02403    2403 5  0,0110 1                TS       SLval
```

```
                          ; verify C(A)
02404    2404 4  0,0000 0                COM                        ; get -A
02405    2405 2  0,0111 1                INDEX    iSL
02406    2406 6  1,2330 0                AD       SLbaseN           ; put (-A) + expected value in A
02407    2407 1  0,0000 0                CCS      A                 ; compare
02410    2410 0  1,2037 1                TC       fail              ; >0 (A < expected value)
02411    2411 0  1,2037 1                TC       fail              ; +0
02412    2412 0  1,2037 1                TC       fail              ; <0 (A > expected value)

                          ; loop back to test next value
02413    2413 1  0,0111 1                CCS      iSL               ; done?
02414    2414 0  1,2377 0                TC       SLloopN           ; not yet, do next check

02415    2415 3  0,0101 1                XCH      savQ
02416    2416 5  0,0001 0                TS       Q                 ; restore return address
02417    2417 0  0,0000 0                RETURN

                          ; ----------------------------------------------
                          ; PASSED ALL TESTS!

02420    2420     12345 0 PASScode       DS       PASS

                          finish         EQU      *
02421    2421 3  1,2420 0                CAF      PASScode
02422    2422 5  0,0100 0                TS       curtest           ; set current test code to PASS
02423    2423 0  0,0000 0                RETURN

                          ; ----------------------------------------------
                          ; INTERRUPT SERVICE ROUTINE

                          goT3           EQU      *
                          goER           EQU      *
                          goDS           EQU      *
                          goKEY          EQU      *
                          goUP           EQU      *


                          endRUPT        EQU      *
02424    2424 3  0,0027 1                XCH      QRUPT             ; restore Q
02425    2425 5  0,0001 0                TS       Q
02426    2426 3  0,0026 0                XCH      ARUPT             ; restore A
02427    2427 2  0,0000 1                RESUME                     ; finished, go back



Assembly complete. Errors = 0

Symbol table:
START          000000   CYRtst         000001   SRtst          000002
CYLtst         000003   SLtst          000004   PASS           012345
EXTENDER       005777   curtest        000100   savQ           000101
CYRval         000102   iCYR           000103   SRval          000104
iSR            000105   CYLval         000106   iCYL           000107
SLval          000110   iSL            000111   GOPROG         002000
T3RUPT         002004   ERRUPT         002010   DSRUPT         002014
KEYRUPT        002020   UPRUPT         002024   goMAIN         002030
fail           002037   end            002041   STRTcode       002042
begin          002043   CYRcode        002046   CYRinit        002047
CYRindx        002050   CYRbase        002051   chkCYR         002070
CYRloop        002077   SRcode         002120   SRinitP        002121
SRinitN        002122   SRindx         002123   SRbaseP        002124
SRbaseN        002143   chkSR          002162   SRloopP        002171
SRloopN        002212   CYLcode        002233   CYLinit        002234
CYLindx        002235   CYLbase        002236   chkCYL         002255
CYLloop        002264   SLcode         002305   SLinitP        002306
SLinitN        002307   SLindx         002310   SLbaseP        002311
SLbaseN        002330   chkSL          002347   SLloopP        002356
SLloopN        002377   PASScode       002420   finish         002421
goT3           002424   goER           002424   goDS           002424
goKEY          002424   goUP           002424   endRUPT        002424
ARUPT          000026   Q              000001   QRUPT          000027
CYR            000020   A              000000   SR             000021
CYL            000022   SL             000023
```

TECO5 assembler listing

```
Block I Apollo Guidance Computer (AGC4) assembler version 1.6 for EPROM

First pass: generate symbol table.
Second pass: generate object code.

                          ; TECO5 (file:teco5.asm)
                          ;
                          ; Version:  1.0
                          ; Author:   John Pultorak
                          ; Date:     9/14/2001
                          ;
                          ; PURPOSE:
                          ; Test and checkout program for the Block 1 Apollo Guidance Computer.
                          ; Tests interrupts.
                          ;
                          ; OPERATION:
                          ; Tests the interrupts by initializing 4 counters to zero and then
                          ; entering a loop where the 1st counter (mainCtr) is incremented on
                          ; each iteration of the loop.
                          ;
                          ; Interrupts are disabled and enabled during each iteration by INHINT
                          ; and RELINT instructions.
                          ;
                          ; Interrupts are automatically inhibited during part of each iteration
                          ; by an overflow condition in register A.
                          ;
                          ; Interrupt service routines for T3RUPT, DSRUPT (aka T4RUPT) and
                          ; KEYRUPT increment their own counters upon each interrupt.
                          ;
                          ; ERRATA:
                          ; - Written for the AGC4R assembler. The assembler directives and
                          ; syntax differ somewhat from the original AGC assembler.
                          ; - The tests attempt to check all threads, but are not exhaustive.
                          ;
                          ; SOURCES:
                          ; Information on the Block 1 architecture: instruction set, instruction
                          ; sequences, registers, register transfers, control pulses, memory and
                          ; memory addressing, I/O assignments, interrupts, and involuntary
                          ; counters was obtained from:
                          ;
                          ;   A. Hopkins, R. Alonso, and H. Blair-Smith, "Logical Description
                          ;           for the Apollo Guidance Computer (AGC4)", R-393,
                          ;           MIT Instrumentation Laboratory, Cambridge, MA, Mar. 1963.
                          ;
                          ; Supplementary information was obtained from:
                          ;
                          ;   R. Alonso, J. H. Laning, Jr. and H. Blair-Smith, "Preliminary
                          ;           MOD 3C Programmer's Manual", E-1077, MIT Instrumentation
                          ;           Laboratory, Cambridge, MA, Nov. 1961.
                          ;
                          ;   B. I. Savage and A. Drake, "AGC4 Basic Training Manual, Volume I",
                          ;           E-2052, MIT Instrumentation Laboratory, Cambridge,
                          ;           MA, Jan. 1967.
                          ;
                          ;   E. C. Hall, "MIT's Role in Project Apollo, Volume III, Computer
                          ;           Subsystem", R-700, MIT Charles Stark Draper Laboratory,
                          ;           Cambridge, MA, Aug. 1972.
                          ;
                          ;   A. Hopkins, "Guidance Computer Design, Part VI", source unknown.
                          ;
                          ;   A. I. Green and J. J. Rocchio, "Keyboard and Display System Program
                          ;           for AGC (Program Sunrise)", E-1574, MIT Instrumentation
                          ;           Laboratory, Cambridge, MA, Aug. 1964.
                          ;
                          ;   E, C. Hall, "Journey to the Moon: The History of the Apollo
                          ;           Guidance Computer", AIAA, Reston VA, 1996.
                          ;

                          ; ----------------------------------------------

                          ; ----------------------------------------------
                          ; ERASEABLE MEMORY -- DATA SEGMENT
```

```
                                         ORG      %47            ; start of data area
00047    0047     00000 1 mainCtr        DS       %0

00050    0050     00000 1 T3Ctr          DS       %0             ; counts T3RUPTs
00051    0051     00000 1 DSCtr          DS       %0             ; counts DSRUPTs (T4RUPT)
00052    0052     00000 1 KYCtr          DS       %0             ; counts KEYRUPT

                          ; ---------------------------------------------
                          ; ENTRY POINTS

                          ; program (re)start
                                         ORG      GOPROG
02000    2000 0  1,2030 0                TC       goMAIN

                          ; interrupt service entry points
                                         ORG      T3RUPT
02004    2004 5  0,0026 0                TS       ARUPT
02005    2005 3  0,0001 0                XCH      Q
02006    2006 5  0,0027 1                TS       QRUPT
02007    2007 0  1,2064 1                TC       goT3

                                         ORG      DSRUPT         ; aka T4RUPT
02014    2014 5  0,0026 0                TS       ARUPT
02015    2015 3  0,0001 0                XCH      Q
02016    2016 5  0,0027 1                TS       QRUPT
02017    2017 0  1,2071 0                TC       goDS

                                         ORG      KEYRUPT
02020    2020 5  0,0026 0                TS       ARUPT
02021    2021 3  0,0001 0                XCH      Q
02022    2022 5  0,0027 1                TS       QRUPT
02023    2023 0  1,2076 1                TC       goKEY


                          ; ---------------------------------------------
                          ; FIXED MEMORY -- SHARED DATA SEGMENT

02024    2024     00000 1 ZERO           DS       %0
02025    2025     00001 0 ONE            DS       %1
02026    2026     25252 0 AD25252        DS       %25252         ;+10922 dec, see TECO1 AD test
02027    2027     52525 1 AD52525        DS       %52525         ;-10922 dec, see TECO1 AD test

                          ; ---------------------------------------------
                          ; MAIN PROGRAM

                          goMAIN         EQU      *
02030    2030 2  0,0000 0                INHINT                  ; disable interrupts

                          ; clear counters for interrupts and for interations
                          ; though main loop.

02031    2031 3  1,2024 0                CAF      ZERO
02032    2032 5  0,0047 1                TS       mainCtr        ; mainCtr = 0
02033    2033 5  0,0050 1                TS       T3Ctr          ; T3Ctr = 0
02034    2034 5  0,0051 0                TS       DSCtr          ; DSCtr = 0
02035    2035 5  0,0052 0                TS       KYCtr          ; KYCtr = 0

                          ; keeps bumping mainCtr in an infinite loop.
                          ; interrupts are disabled and enabled on each
                          ; iteration of the loop.

                          infLoop        EQU      *

02036    2036 2  0,0000 0                INHINT                  ; disable interrupt

                          ; increment mainCtr while interrupt is inhibited.

02037    2037 3  1,2024 0                CAF      ZERO
02040    2040 6  0,0047 1                AD       mainCtr        ; load mainCtr into A
02041    2041 6  1,2025 1                AD       ONE            ; incr

02042    2042 2  0,0000 1                RELINT                  ; enable interrupts

02043    2043 5  0,0047 1                TS       mainCtr        ; store increment value
```

```
                          ; create a positive overflow in A. Interrupts are inhibited
                          ; while A contains an overflow. The overflow is produced
                          ; by adding %25252 + %25252 = %52524 (sign + 14 magnitude).
                          ; This is the overflow test in TECO1 for the AD instruction.

02044    2044 3  1,2026 1                CAF      AD25252        
02045    2045 6  1,2026 1                AD       AD25252        ; positive overflow

02046    2046 3  0,0000 1                NOOP                    ; interrupt should be inhib
02047    2047 3  0,0000 1                NOOP

                          ; remove the overflow, this reenables the interrupt.

02050    2050 3  1,2024 0                CAF      ZERO           ; clear the overflow in A

02051    2051 3  0,0000 1                NOOP                    ; interrupt should be reenab
02052    2052 3  0,0000 1                NOOP

                          ; create a negative overflow in A. Interrupts are inhibited
                          ; while A contains an overflow. The overflow is produced
                          ; by adding %52525 + %52525 = %25253 (sign + 14 magnitude).
                          ; This is the overflow test in TECO1 for the AD instruction.

02053    2053 3  1,2027 0                CAF      AD52525
02054    2054 6  1,2027 0                AD       AD52525        ; positive overflow

02055    2055 3  0,0000 1                NOOP                    ; interrupt should be inhib
02056    2056 3  0,0000 1                NOOP

                          ; remove the overflow, this reenables the interrupt.

02057    2057 3  1,2024 0                CAF      ZERO           ; clear the overflow in A

02060    2060 3  0,0000 1                NOOP                    ; interrupt should be reenab
02061    2061 3  0,0000 1                NOOP


02062    2062 0  1,2036 0                TC       infLoop        ; mainCtr no overflow
02063    2063 0  1,2036 0                TC       infLoop        ; mainCtr overflowed

                          ; ------------------------------------------------
                          ; INTERRUPT SERVICE ROUTINE

                          goT3           EQU      *
02064    2064 3  1,2024 0                CAF      ZERO
02065    2065 6  0,0050 1                AD       T3Ctr          ; load T3Ctr into A
02066    2066 6  1,2025 1                AD       ONE            ; incr
02067    2067 5  0,0050 1                TS       T3Ctr          ; store
02070    2070 0  1,2103 1                TC       endRUPT

                          goDS           EQU      *
02071    2071 3  1,2024 0                CAF      ZERO
02072    2072 6  0,0051 0                AD       DSCtr          ; load DSCtr into A
02073    2073 6  1,2025 1                AD       ONE            ; incr
02074    2074 5  0,0051 0                TS       DSCtr          ; store
02075    2075 0  1,2103 1                TC       endRUPT

                          goKEY          EQU      *
02076    2076 3  1,2024 0                CAF      ZERO
02077    2077 6  0,0052 0                AD       KYCtr          ; load KYCtr into A
02100    2100 6  1,2025 1                AD       ONE            ; incr
02101    2101 5  0,0052 0                TS       KYCtr          ; store
02102    2102 0  1,2103 1                TC       endRUPT

                          endRUPT        EQU      *
02103    2103 3  0,0027 1                XCH      QRUPT          ; restore Q
02104    2104 5  0,0001 0                TS       Q
02105    2105 3  0,0026 0                XCH      ARUPT          ; restore A
02106    2106 2  0,0000 1                RESUME                  ; finished, go back



Assembly complete. Errors = 0

Symbol table:
```

```
mainCtr        000047   T3Ctr          000050   DSCtr          000051
KYCtr          000052   GOPROG         002000   T3RUPT         002004
DSRUPT         002014   KEYRUPT        002020   ZERO           002024
ONE            002025   AD25252        002026   AD52525        002027
goMAIN         002030   infLoop        002036   goT3           002064
goDS           002071   goKEY          002076   endRUPT        002103
ARUPT          000026   Q              000001   QRUPT          000027
```

TECO_STBY assembler listing

```
Block I Apollo Guidance Computer (AGC4) assembler version 1.6

First pass: generate symbol table.
Second pass: generate object code.

                          ; TECO_STBY (file:stby.asm)
                          ;
                          ; Tests the standby operation.

                          ; program (re)start
                                         ORG      GOPROG
02000    2000 0  1,2002 1                TC       goMAIN

02001    2001     00200 0 ofbit          DS       %200           ; OUT1, bit 8 initiates standby


                          ; MAIN PROGRAM

                          goMAIN         EQU      *

                          ; standby is disabled
02002    2002 3  0,0000 1                NOOP
02003    2003 3  0,0000 1                NOOP

                          ; enable standby
02004    2004 3  1,2001 1                XCH      ofbit
02005    2005 5  0,0011 1                TS       OUT1

                          infLoop        EQU      *
02006    2006 0  1,2006 0                TC       infLoop



Assembly complete. Errors = 0

Symbol table:
GOPROG         002000   ofbit          002001   goMAIN         002002
infLoop        002006   OUT1           000011
```