EDSYS29

INTRODUCTION TO THE AMC SYS/29 DEVELOPMENT SYSTEM

LABS & EXERCISES

AMD CUSTOMER EDUCATION CEC $15.00

Publication number: CEC.PUB-29-10
2nd Edition
February 10, 1981

901 Thompson Place, P.O. Box 453, Sunnyvale, California 94086
(408)732-2400 TWX: 910-339-9280 TELEX: 34-6306

EDSYS29

STUDY GUIDE AND LAB BOOK

INCLUDING EXAMPLE PROGRAM LISTINGS

for the Advanced Micro Computers

AmSYS29

by

Donnamaie E. White

Advanced Micro Devices, Inc.

Customer Education Center

2nd Edition

February, 1981

PREFACE

This lab book was designed to accompany the Advanced Micro Devices, Inc. seminar/workshop on the Advanced Micro Computers AmSYS29 Development System. The seminar concentrates on the AmSYS29 as a development tool for 2900 Family designs and as such emphasizes the microprogram development tools. It does not cover the use of the system for 8080, 8085, Z80, or Z8000 software/hardware development.

The seminar was designed to follow the ED2900A "Introduction to Design with the 2900 Family" seminar, which introduces the 2900 family and also introduces the concept of structured microprogramming, including the use of mnemonics. This includes the concepts behind the .DEF and .SRC AMDASM files.

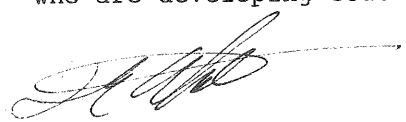
For those who have not taken this prerequisite, the initial labs may be more difficult. For this reason, the problems from ED2900A are included in the lab book for study and review. The complete solution to the coffee machine problem is included. The instructor facing a class, where most or all of the students have not taken ED2900A (or the new ED2910), should very carefully go over the problem, the manual solution,and then refer to the .DEF and .SRC files as examples during the lecture.

The instructor and student should be sure to have the correct data disk to go with the labs, since the labs will refer to specific files. If such a disk is not available or cannot be obtained in time, the disk can be created from the sample programs that are included throughout the lab book.

This seminar takes the student up to interfacing to a prototype, but does not include actual driving of a prototype from the writeable control store. The ED2900B "Microprogrammable Computer Architecture" or "Advanced Design with the 2900 Family" seminar does include labs wherein the AmSYS29 is interconnected to a HEX-29 minicomputer. EDSYS29 is a prerequisite for ED2900B. ED2900C is the non-lab (non-HEX-29) version of ED2900B.

None of the labs include interconnection to a PROM-burner, but the Data-I/O application note on such interconnection is included.

Two reference .DEF files are included, which should be mentioned. AM2900.LIB is an older file which was created to provide users with a file from which they could patch-up a .DEF file with a minimum of effort. It contains both older and newer devices, and is not as easy to use as it might be. Included also is the AM2903.DEF file, which includes a new .DEF file for the Am2903, the Am2910, and a partial for the Am2904. (The Am2904 uses bit-steering and it would be prohibitive to list all possibilities.) This file is referenced in ED2900A and ED2900B/C. It is considerably easier to use, especially for those who are new to the 2900 family or the AmSYS29 system. By altering the definition statements and the word width, and adding those equates and other statements as needed, this file can be used by those who are developing code for an Am2903-based design.



Donnamaie E. White, Ph.D.

STUDY GUIDE INDEX

INDEX

DAY 1

****

STUDY GUIDE INDEX

DAY 2

LAB ONE:

GETTING ACQUAINTED WITH

THE AMC SYS/29

- FILE HANDLING

LAB ONE - INTRODUCTION TO FILE HANDLING

RULES OF THE GAME

1. BREAK UP INTO TEAMS - USE YOUR HOTELS, HOME ADDRESS, OR OTHER DATA TO CHOOSE PARTNERS YOU WILL MAINTAIN THROUGHOUT THE CLASS

2. THOSE OF YOU WITH 2900 FAMILY DESIGN EXPERIENCE SHOULD BE SPREAD OUT AMONG THE OTHERS

3. TRY TO DISTRIBUTE HARDWARE AND SOFTWARE EXPERTISE AMONG THE TEAMS

4. HOMEWORK AND LAB SESSIONS ARE TO BE A GROUP EFFORT - YOU WILL LEARN MORE THAT WAY!

PURPOSE OF THE LAB: FAMILIARIZATION WITH BASIC FILE HANDLING FUNCTIONS OF AMC SYSTEM 29.

LAB STATIONS

EACH LAB STATION CONSISTS OF:

- ONE AmSYS 29 CPU BOX - FRONT PANEL SWITCHES
  NOTE: THERE IS ANOTHER MODEL WHICH
  LOOKS SLIGHTLY DIFFERENT - WORKS THE SAME
- ONE LINE PRINTER
- ONE CRT - EITHER ADDS OR THE OLDER TELERAY
- ONE DUAL 8" FLOPPY DISK DRIVE SYSTEM

CHECK TO SEE THAT YOUR TEAM HAS THE FOLLOWING:

- A SYSTEM DISK LABELED VER 1.4
- A DATA DISK (WHICH IS NOT BLANK)

DO NOT MISHANDLE THE DISKETTES!

- DO NOT FOLD, SPINDLE OR MUTILATE
- NO COFFEE, SUGAR OR CREAM
- NO SODA, MAYONNAISE, ETC.
- DO NOT WRITE WITH A PENCIL OR A BALL POINT PEN

CHECK TO SEE THAT THERE IS PAPER FEEDING INTO THE PRINTER

- FEEDING STRAIGHT IN
- ON THE TRACTOR PINS
- PRINT IS CLEAR (MAY NOT BE SHARP BLACK BUT SHOULD BE READABLE)
- DO NOT STEP ON, PLACE CHAIRS ON OR OTHERWISE BLOCK PAPER FEED

NORMAL SYSTEM POWER-UP

IF YOU OWN AN AmSYS29 THERE IS A CONVENTIONAL METHOD OF POWER-ON AND SIGN-ON

WE WILL BYPASS THIS

LAB POWER-ON PROCEDURE

- MAKE SURE THAT ALL UNITS ARE PROPERLY PLUGGED IN
- MAKE CERTAIN THAT THE CRT IS CONNECTED TO THE BACK OF THE SYSTEM 29 CPU
- MAKE CERTAIN THAT THE PRINTER IS CONNECTED
- MAKE CERTAIN THAT THE FLOPPY DISK SYSTEM IS CONNECTED
- TURN ON ALL SWITCHES

2 SWITCHES ON FRONT OF SYSTEM 29 CPU
CHECK ALL OTHER SWITCHES ON THE CPU

1 DIAL SWITCH ON THE CRT (TELERAY)
OR SWITCH ON RIGHT BOTTOM OF ADDS TERMINAL

1 SWITCH ON THE FRONT OF THE DISK DRIVE

1 ON THE LOWER LEFT REAR OF THE PRINTER

DISKETTE INSERTION

- PLACE THE SYSTEM VER 1.4 DISKETTE IN THE RIGHT SIDE OF THE DISK UNIT, THIS IS DRIVE "A" (NEWER SYSTEMS STACK THE DRIVES, THE TOP UNIT IS DRIVE A)

  BE SURE THAT THE DISKETTE IS HELD BY THE LABEL EDGE

  BE SURE THE LABEL EDGE EXTENDS TO THE FRONT

  BE SURE THAT THE LABEL IS FACING UP WHEN PLACING THE DISKETTE INTO THE DRIVE

  INSERT FIRMLY BUT DO NOT BEND

  CLOSE DOOR - BUT NOT ON DISKETTE!

- PRESS "RESET" SWITCH ON THE FRONT OF THE AmSYS/29 CPU

YOU SHOULD HEAR SOME ACTIVITY FROM THE DISK UNIT AND SEE SOME MESSAGE ON THE SCREEN

IF YOU DO, YOU'RE OK

IF YOU DO NOT, YOU'RE NOT BOOTING UP

SIGN ON

* ONCE BOOTED, TYPE THE "CTRL" KEY AND THE Z KEY AT THE SAME TIME (Zc) THEN PRESS THE "RETURN" KEY (RET) WHICH IS LABELED AS "NEW LINE" ON THE ADDS TERMINAL

* THE SYSTEM WILL RESPOND WITH

A>_

- A> REFERS TO THE ACTIVE OR SIGNED-ON DRIVE
- THE _ MARKS THE CURSOR POSITION
- THE > IS CONSIDERED TO BE THE SYSTEM PROMPT - DIFFERENT SYSTEMS USE DIFFERENT SYMBOLS AS PROMPTS (YOU WILL BE SEEING . > *)

* INSERT YOUR DATA DISKETTE IN DRIVE "B" (LEFT HAND SIDE)

ANY TIME THAT A DISKETTE IS CHANGED YOU MUST LET THE SYSTEM KNOW BY TYPING Cc AND "RETURN" (Ret) (THIS IS A "WARM START")

TYPE: Cc AND Ret AND THEN LISTEN (YOU SHOULD HEAR SOME ACTIVITY FROM THE DRIVE)

COMMAND SUMMARY

DIR	DIRECTORY, LISTING OF FILES
Pc	ENABLE/DISABLE PRINTER
STAT	DISKETTE STATUS
PIP	PERIPHERAL INTERCHANGE PROGRAM
	(SAY IT THREE TIMES QUICKLY!)
ERA	ERASE FILES
TYPE	PRINT A FILE TO SCREEN
	(EVELYN WOOD SPEED READING!)
	USE WHEN PRINTER IS ENABLED TO OBTAIN A LISTING
Cc	WARM START
DISPL	DISPLAY A FILE PAGE AT A TIME
	USE WHEN PRINTER DISABLED, FOR HUMAN-CRT DISPLAY

LAB EXERCISES

- TYPE THE FOLLOWING:

DIR(Ret) (I WILL USE (Ret) TO MEAN RETURN KEY)

- THIS IS A REQUEST TO THE SYSTEM TO LIST THE DIRECTORY OF THE DISKETTE CURRENTLY ON THE SIGN-ON DRIVE (DRIVE "A")

DIRPc

- THE Pc TURNS ON THE PRINTER AND THE DIRECTORY WILL NOW APPEAR BOTH ON THE CRT SCREEN AND ON THE PRINTER

Pc

- THE PRINTER SHOULD BECOME QUIET

B:(Ret)

- THIS LOGS ON DRIVE "B" AND THE SCREEN SHOULD BE DISPLAYING

B>_

DIR(Ret)

- THIS TIME NOTE THE B> WHICH PRECEEDS THE FILES LISTED

DO YOU SEE COFFEE.DEF, COFFEE.SRC ?
YOU SHOULD ALSO SEE FILES WITH THE
PRIMARY NAMES OF:
MONITOR
SIMPLE
AM2903

DIR A:(Ret)

- WHILE DEFAULT IS TO THE LOGGED ON DRIVE, IT CAN BE OVERRIDDEN BY SPECIFICATION OF THE DESIRED DRIVE

A:(Ret)

- TYPE THE FOLLOWING IN ORDER (ASSUME (Ret) AT THE END OF EACH)

STAT

STATUS REQUEST

STAT *.*

STAT B:*.*

● CONTINUE TYPING

```
TYPE B:SIMPLE.DEF          <----  TYPE THE WORD "TYPE"
        USED WITH Pc WILL ALLOW PRINT OUT OF A FILE



DISPL B:SIMPLE.DEF
          THIS IS A PAGED FILE DISPLAY
          HIT (Ret) UNTIL "*" APPEARS
          TWICE AT THE BOTTOM OF THE SCREEN

E(Ret)
          E STANDS FOR EXIT

DIR B:SIMPLE.*
        THESE CHECK TO FIND A FILE IN A DIRECTORY
```

● TURN PRINTER ON: Pc

```
TYPE B:COFFEE.DEF

TYPE B:COFFEE.SRC

Pc
```

CONTINUE:

AMDASM P1 B:COFFEE P2 B:COFFEE
DO NOT PANIC! THE DISKS ARE NOISY!

Pc
TYPE B:COFFEE.P2L
Pc

PIP B:SIMPLE.TST=B:SIMPLE.DEF[V]

PERIPHERAL INTERCHANGE PROGRAM "COPY"
PIP IS ALWAYS "TO FILE" = "FROM FILE"
NOTE: YOU SHOULD CLOSE THE BRACKET
(IT WILL ACCEPT (Ret) AS A TERMINATOR)

DIR B:SIMPLE.*

PIP B:TESTONE=A:SIMPLE.DEF[V]

DIR B:TESTONE

TYPE B:TESTONE

TYPE Sc IMMEDIATELY TO HALT SCREEN
TYPE (Ret) OR Sc TO CONTINUE

ERA B:TESTONE

PIP B:=A:AM2903.DEF[V]

● TYPE THE FOLLOWING AND FOLLOW BY (Ret)

PIP
LST:=AM2903.DEF } or { PIP LST:=AM2903.DEF
(Ret) or Cc or { PIP PRN:=AM2903.DEF

PIP B:LIGHT.DEF=A:AM2903.DEF[VSJMAPZcQTWBZcT8]

● MOVE IS INCLUSIVE

TYPE B:LIGHT.DEFPc

Pc

END OF EXERCISE

* REMOVE DISKETTES

* POWER DOWN

EXERCISES:

FILENAMES

AMDASM

EXERCISES

ARE THESE PROPER FILE NAMES OR FILE NAME REFERENCES?

____DOOR.DEF
____DOOR.*
____B:*.DEF
____D?OR.D?F
____123.456
____TEMP
____DOOR.SRC
____B:DOOR.SRC
____*.*
____B:X?X.*
____1-2-3
____B:TEMP

WHAT EXTENSION NAME IS REQUIRED FOR THE DEFINITION FILE BEFORE IT CAN BE ASSEMBLED VIA AMDASM?

WHAT EXTENSION NAME IS REQUIRED FOR THE SOURCE FILE BEFORE IT CAN BE ASSEMBLED BY AMDASM?

IS THE EXTENSION REQUIRED WHEN CALLING FOR AN ASSEMBLY?

EXERCISES

WHAT SYMBOL STARTS A COMMENT?

WHAT SYMBOL STARTS A LINE THAT IS A CONTINUATION OF THE PRECEEDING LINE?

WHAT IS THE DESIGNATOR FOR A HEX CONSTANT?

HOW MANY CHARACTERS CAN BE IN A VARIABLE NAME?

WHAT CHARACTERS MAY BE THE FIRST CHARACTER IN A VARIABLE NAME?

WHAT DETERMINES IF % IS A MODIFIER OR AN ATTRIBUTE?

WHAT IS THE ATTRIBUTE $ EQUIVALENT TO?

IF NO BASE IS GIVEN IN AN EQU STATEMENT, WHAT IS THE DEFAULT?

IF NO BASE IS GIVEN IN A DEF STATEMENT, WHAT IS THE DEFAULT?

IF NO BASE IS GIVEN IN AN ASSEMBLY STATEMENT VARIABLE FIELD SUBSTITUTION, WHAT IS THE DEFAULT?

CAN EQU STATEMENTS APPEAR IN THE DEF AND SRC FILES?

CAN DEF STATEMENTS APPEAR IN THE DEF AND SRC FILES?

WHAT IS THE STATEMENT WORD USED FOR?

HOW WIDE CAN A VARIABLE FIELD BE (NUMBER OF BITS)?

HOW WIDE CAN A DON'T CARE FIELD BE?

WHAT IS THE MAXIMUM NUMBER OF FIELDS ALLOWED IN A DEF STATEMENT?

THE COFFEE MACHINE REVISITED

DEF & SRC FILES

You are to design a coffee machine controller that will handle a simple, non-fault diagnostic coffee dispenser. It will work as follows:

1. Do nothing until a coin is detected.
2. On coin detection, turn on the busy light and drop a cup.
3. The cup has 1.5 seconds to get into place.
4. There is no way to know if the cup is correctly positioned or if it even is there.
5. Water is turned on for 1.0 second prior to the release of powders (so it isn't unsightly in the bottom of the cup).
6. Water will remain on continuously for a total of 10 seconds.
7. The busy light will remain on until the sequence is completed.
8. Depending upon the selection, either coffee, soup, or chocolate will be dispensed.

coffee	2.5 seconds
soup	2.0 seconds
chocolate	3.5 seconds

9. If coffee was selected, when the coffee is finished, sugar or cream is dispensed.

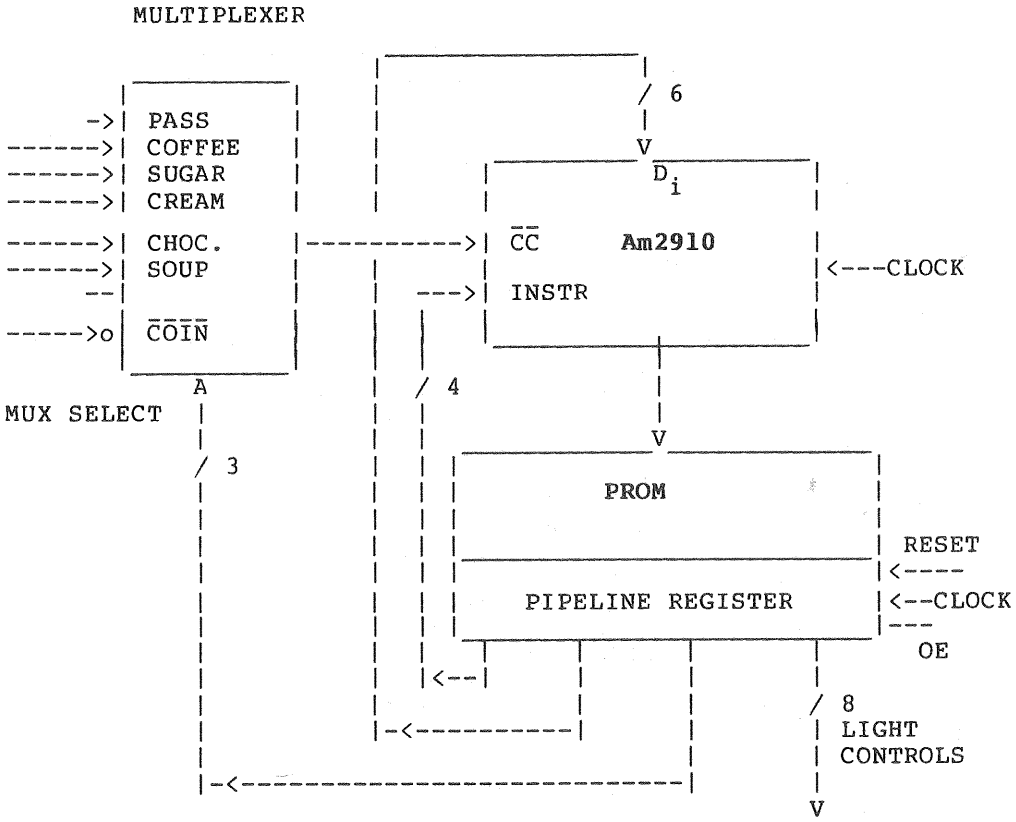
sugar	1.5 seconds
cream	2.0 seconds

10. If sugar and cream was the selection, when the sugar is finished, start cream.
11. After the water has completed filling the cup, allow 3.5 seconds for cup removal before testing for the presence of the next coin.
12. You have a 0.5 second clock pulse.

There are six possible sequences:

COFFEE, BLACK
COFFEE, CREAM
COFFEE, SUGAR
COFFEE, CREAM AND SUGAR
CHOCOLATE
SOUP

Am2910 HARDWARE SOLUTION



A = COFFEE B = CREAM C = SUGAR

MUX INPUT:
COFFEE-ON <=> A + AB + AC + ABC
CREAM-ON <=> AB + ABC
SUGAR-ON <=> AC + ABC

THE MICROPROGRAM

LABEL	ADDR	Am2910 INSTR.	COND. MUX	BRANCH COUNTER	MACHINE CONTROL
ZERO	0	CJP	$\overline{\text{COIN}}$	ZERO	OFF
	1	CONT	#	#	CUP-BUSY
	2	CONT	#	#	BUSY
	3	CONT	#	#	BUSY
	4	CJP	CHOC	CHOC	WATER-BUSY
	5	CJP	SOUP	SOUP	WATER-BUSY
COFFEE	6	LDCT	#	12	COFFEE-WTR-BUSY
	7	CONT	#	#	COFFEE-WTR-BUSY
	8	CONT	#	#	COFFEE-WTR-BUSY
	9	CJP	SUGAR	SUGAR	COFFEE-WTR-BUSY
	10	CJP	CREAM	CRE-2	COFFEE-WTR-BUSY
LOOP	11	RPCT	#	LOOP	WATER-BUSY
	12	LDCT	#	4	BUSY
BUSY	13	RPCT	#	BUSY	BUSY
	14	JZ	#	ZERO	BUSY
SUGAR	15	CONT	#	#	COFFEE-WTR-BUSY
	16	LDCT	#	9	SUGAR-WTR-BUSY
	17	CJP	CREAM	CREAM	SUGAR-WTR-BUSY
	18	CJP	PASS	LOOP	SUGAR-WTR-BUSY
CREAM	19	LDCT	#	5	SUGAR-WTR-BUSY
	20	CONT	#	#	CREAM-WTR-BUSY
	21	CONT	#	#	CREAM-WTR-BUSY
ENTRY	22	CONT	#	#	CREAM-WTR-BUSY
	23	CJP	PASS	LOOP	CREAM-WTR-BUSY
CRE-2	24	LDCT	#	8	CREAM-WTR-BUSY
	25	CJP	PASS	ENTRY	CREAM-WTR-BUSY

```
CHOC     26      CONT      #        #        WATER-BUSY
         27      LDCT      #        5        CHOC-WTR-BUSY
CHOCLP   28      RPCT      #        CHOCLP   CHOC-WTR-BUSY
         29      LDCT      #        8        WATER-BUSY
         30      CJP       PASS     LOOP     WATER-BUSY
-------------------------------------------------------------

SOUP     31      CONT      #        #        SOUP-WTR-BUSY
         32      CONT      #        #        SOUP-WTR-BUSY
         33      LDCT      #        13       SOUP-WTR-BUSY
         34      CJP       PASS     LOOP     SOUP-WTR-BUSY
-------------------------------------------------------------

          :                                    .
-------------------------------------------------------------

         63      JZ        #        #        OFF
-------------------------------------------------------------
```

COFFEE MACHINE .DEF and .SRC

THE FAMOUS COFFEE MACHINE

- COFFEE.DEF listing
- COFFEE.SRC listing
- COFFEE.P2L AMDASM ASSEMBLY listing
    - .SRC SEQUENCED
    - CONTROL MEMORY PRINTOUT (X for Don't Cares)
    - SYMBOLS list

```
TITLE COFFEE MACHINE
;
WORD 21
;                                                    .DEF
;AM2910 INSTRUCTIONS
;
JZ:       EQU       H#0       ;JUMP TO ZERO
CJP:      EQU       H#3       ;;CONDITIONAL JUMP
RPCT:     EQU       H#9       ;DO-LOOP
LDCT:     EQU       H#C       ;LOAD COUNTER AND CONTINUE
CONT:     EQU       H#E       ;CONTINUE
;
;CONDITIONAL MUX SELECT FIELD
;
NOCOIN:   EQU       Q#0       ;TEST FOR COIN
NULL:     EQU       Q#1
SOUPTST:  EQU       Q#2
CHOCTST:  EQU       Q#3
CREMTST:  EQU       Q#4
SUGRTST:  EQU       Q#5
CAFETST:  EQU       Q#6
PASS:     EQU       Q#7       ;ACTIVE LOW PASS
;
;MACHINE CONTROLS
;
; THE BIT LAYOUT PATTERN IS
; BUSY(LIGHT)-CUP(DROP)-WATER-COFFEE-SUGAR-CREAM-CHOCOLATE-SOUP
;
OFF:      EQU       H#00
BUSYON:   EQU       H#80
CUPDROP:  EQU       H#C0
WATERON:  EQU       H#A0
COFFEON:  EQU       H#B0
SUGARON:  EQU       H#A8
CREAMON:  EQU       H#A4
CHOCON:   EQU       H#A2
SOUPON:   EQU       H#A1
;
;FORMAT DEFINITION
;
MACHINE:  DEF       13X,      8VH#00              ;MACHINE CONTROLS
SEQ:      DEF       4VH#E,    3VQ#1, 6V$X,8X      ;NEXT ADDRESS CTRL
;
END
```

```
TITLE COFFEE MACHINE SRC FILE                                    .SRC
;
;LABEL
;-------------------------------------------------------------------------
ZERO:    SEQ CJP, NOCOIN, ZERO     & MACHINE OFF  ; WAIT FOR COIN
         SEQ                       & MACHINE CUPDROP
         SEQ                       & MACHINE BUSYON
         SEQ                       & MACHINE BUSYON
;
; TURN ON WATER AND TEST TO FIND ROUTINE
;
         SEQ CJP, CHOCTST, CHOC    & MACHINE WATERON
         SEQ CJP, SOUPTST, SOUP    & MACHINE WATERON
;-------------------------------------------------
;
; COFFEE BUTTON HAS BEEN PUSHED - DISPENSE COFFEE
;
COFFEE:  SEQ LDCT,         , H#C   & MACHINE COFFEON
         SEQ                       & MACHINE COFFEON
         SEQ                       & MACHINE COFFEON
         SEQ CJP,SUGRTST, SUGAR    & MACHINE COFFEON
         SEQ CJP,CREMTST, CRM2     & MACHINE COFFEON
;-------------------------------------------------
;
; FINISH FILLING THE CUP WITH WATER
;
LOOP:    SEQ RPCT,         , LOOP  & MACHINE WATERON
         SEQ LDCT,         , H#4   & MACHINE BUSYON
;-------------------------------------------------
;
; ALLOW TIME TO REMOVE CUP BEFORE STARTING OVER
;
BUSY:    SEQ RPCT,         , BUSY  & MACHINE BUSYON
         SEQ JZ                    & MACHINE BUSYON
;-------------------------------------------------
;
; DISPENSE SUGAR
;
SUGAR:   SEQ                         & MACHINE COFFEON
         SEQ LDCT,         , H#9     & MACHINE SUGARON
         SEQ CJP,CREMTST, CREAM      & MACHINE SUGARON
         SEQ CJP, PASS   , LOOP      & MACHINE SUGARON
;-------------------------------------------------
;
; DISPENSE CREAM (COFFEE, SUGAR AND CREAM ENTRY)
;
CREAM:   SEQ LDCT,         , H#5     & MACHINE SUGARON
         SEQ                       & MACHINE CREAMON
         SEQ                       & MACHINE CREAMON
ENTRY:   SEQ                       & MACHINE CREAMON
         SEQ CJP, PASS   , LOOP    & MACHINE CREAMON
;-------------------------------------------------
;
; DISPENSE CREAM (COFFEE AND CREAM ENTRY)
;
```

```
CRM2:   SEQ LDCT,       , H#8       & MACHINE CREAMON
        SEQ CJP, PASS   , ENTRY     & MACHINE CREAMON
;-------------------------------------------------
;
; DISPENSE CHOCOLATE
;
CHOC:   SEQ                         & MACHINE WATERON
        SEQ LDCT,       , H#5       & MACHINE CHOCON
CHOCLP: SEQ RPCT,       , CHOCLP    & MACHINE CHOCON
        SEQ LDCT,       , H#8       & MACHINE WATERON
        SEQ CJP, PASS   , LOOP      & MACHINE WATERON
;-------------------------------------------------
;
; DISPENSE SOUP
;
SOUP:   SEQ                         & MACHINE SOUPON
        SEQ                         & MACHINE SOUPON
        SEQ LDCT,       , H#D       & MACHINE SOUPON
        SEQ CJP, PASS   , LOOP      & MACHINE SOUPON
;-------------------------------------------------
;
; SKIP OVER MEMORY WHICH IS UNUSED
;
ORG 63
        SEQ JZ                      & MACHINE
;
END
```

NOTICE THAT

IF THE DEFINITION OF THE FORMAT HAD BEEN:

```
SEQ: DEF 4VH#E, 3VQ#1, 6V$X, 8VH#00
```

THEN THE SRC FILE WOULD HAVE BEEN ALTERED TO BE:

```
COFFEE: SEQ LDCT,      , H#C, COFFEON
        SEQ     ,      ,     , COFFEON

BUSY:   SEQ RPCT,      , BUSY, BUSYON

        etc.
```

WHICH METHOD IS CORRECT?

BOTH ARE

WHICH SHOULD YOU USE?

EITHER

```
      TITLE COFFEE MACHINE SRC FILE
      ;
      ;LABEL   COFFEE MACHINE      PHASE 2
      ;-------------------------------------------------
0000  ZERO:    SEQ CJP, NOCOIN, ZERO     & MACHINE OFF  ; WAIT FOR COIN
0001           SEQ                       & MACHINE CUPDROP
0002           SEQ                       & MACHINE BUSYON
0003           SEQ                       & MACHINE BUSYON
      ;
      ; TURN ON WATER AND TEST TO FIND ROUTINE
      ;
0004           SEQ CJP, CHOCTST, CHOC    & MACHINE WATERON
0005           SEQ CJP, SOUPTST, SOUP    & MACHINE WATERON
      ;-------------------------------------------------
      ;
      ; COFFEE BUTTON HAS BEEN PUSHED - DISPENSE COFFEE
      ;
0006  COFFEE:  SEQ LDCT,        , H#C     & MACHINE COFFEON
0007           SEQ                       & MACHINE COFFEON
0008           SEQ                       & MACHINE COFFEON
0009           SEQ CJP,SUGRTST, SUGAR    & MACHINE COFFEON
000A           SEQ CJP,CREMTST, CRM2     & MACHINE COFFEON
      ;-------------------------------------------------
      ;
      ; FINISH FILLING THE CUP WITH WATER
      ;
000B  LOOP:    SEQ RPCT,        , LOOP    & MACHINE WATERON
000C           SEQ LDCT,        , H#4     & MACHINE BUSYON
      ;-------------------------------------------------
      ;
      ; ALLOW TIME TO REMOVE CUP BEFORE STARTING OVER
      ;
000D  BUSY:    SEQ RPCT,        , BUSY    & MACHINE BUSYON
000E           SEQ JZ                    & MACHINE BUSYON
      ;-------------------------------------------------
      ;
      ; DISPENSE SUGAR
      ;
000F  SUGAR:   SEQ                       & MACHINE COFFEON
0010           SEQ LDCT,        , H#9     & MACHINE SUGARON
0011           SEQ CJP,CREMTST, CREAM     & MACHINE SUGARON
0012           SEQ CJP, PASS    , LOOP    & MACHINE SUGARON
      ;-------------------------------------------------
      ;
      ; DISPENSE CREAM (COFFEE, SUGAR AND CREAM ENTRY)
      ;
0013  CREAM:   SEQ LDCT,        , H#5     & MACHINE SUGARON
0014           SEQ                       & MACHINE CREAMON
0015           SEQ                       & MACHINE CREAMON
0016  ENTRY:   SEQ                       & MACHINE CREAMON
0017           SEQ CJP, PASS    , LOOP    & MACHINE CREAMON
      ;-------------------------------------------------
```

```
     ; DISPENSE CREAM (COFFEE AND CREAM ENTRY)
     ;
0018 CRM2:   SEQ LDCT,       , H#8       & MACHINE CREAMON
0019         SEQ CJP, PASS   , ENTRY     & MACHINE CREAMON
     ;-------------------------------------------------
     ;
     ; DISPENSE CHOCOLATE
     ;
001A CHOC:   SEQ                         & MACHINE WATERON
001B         SEQ LDCT,       , H#5       & MACHINE CHOCON
001C CHOCLP: SEQ RPCT,       , CHOCLP    & MACHINE CHOCON
001D         SEQ LDCT,       , H#8       & MACHINE WATERON
001E         SEQ CJP, PASS   , LOOP      & MACHINE WATERON
     ;-------------------------------------------------
     ;
     ; DISPENSE SOUP
     ;
001F SOUP:   SEQ                         & MACHINE SOUPON
0020         SEQ                         & MACHINE SOUPON
0021         SEQ LDCT,       , H#D       & MACHINE SOUPON
0022         SEQ CJP, PASS   , LOOP      & MACHINE SOUPON
     ;-------------------------------------------------
     ;
     ; SKIP OVER MEMORY WHICH IS UNUSED
     ;
003F ORG 63
003F         SEQ JZ                      & MACHINE
     ;
     END
```

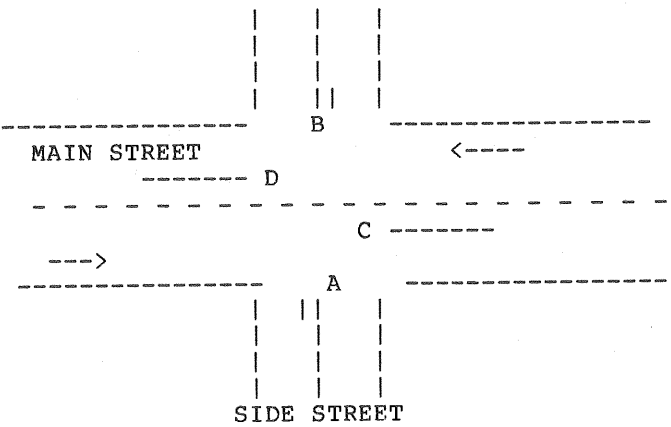
```
MDOS/29 AMDASM MICRO ASSEMBLER, V1.4
COFFEE MACHINE SRC FILE

 0000 0011000000000000 00000
 0001 1110001XXXXXX110 00000
 0002 1110001XXXXXX100 00000
 0003 1110001XXXXXX100 00000
 0004 0011011011010101 00000
 0005 0011010011111101 00000
 0006 1100001001100101 10000
 0007 1110001XXXXXX101 10000
 0008 1110001XXXXXX101 10000
 0009 0011101001111101 10000
 000A 0011100011000101 10000
 000B 1001001001011101 00000
 000C 1100001000100100 00000
 000D 1001001001101100 00000
 000E 0000001XXXXXX100 00000
 000F 1110001XXXXXX101 10000
 0010 1100001001001101 01000
 0011 0011100010011101 01000
 0012 0011111001011101 01000
 0013 1100001000101101 01000
 0014 1110001XXXXXX101 00100
 0015 1110001XXXXXX101 00100
 0016 1110001XXXXXX101 00100
 0017 0011111001011101 00100
 0018 1100001001000101 00100
 0019 0011111010110101 00100
 001A 1110001XXXXXX101 00000
 001B 1100001000101101 00010
 001C 1001001011100101 00010
 001D 1100001001000101 00000
 001E 0011111001011101 00000
 001F 1110001XXXXXX101 00001
 0020 1110001XXXXXX101 00001
 0021 1100001001101101 00001
 0022 0011111001011101 00001
 003F 0000001XXXXXX000 00000
```

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.0
COFFEE MACHINE SRC FILE

 SYMBOLS

  BUSY      000D
  BUSYON    0080
  CAFETST   0006
  CHOC      001A
  CHOCLP    001C
  CHOCON    00A2
  CHOCTST   0003
  CJP       0003
  COFFEE    0006
  COFFEON   00B0
  CONT      000E
  CREAM     0013
  CREAMON   00A4
  CREMTST   0004
  CRM2      0018
  CUPDROP   00C0
  ENTRY     0016
  JZ        0000
  LDCT      000C
  LOOP      000B
  NOCOIN    0000
  NULL      0001
  OFF       0000
  PASS      0007
  RPCT      0009
  SOUP      001F
  SOUPON    00A1
  SOUPTST   0002
  SUGAR     000F
  SUGARON   00A8
  SUGRTST   0005
  WATERON   00A0
  ZERO      0000

 TOTAL PHASE 2 ERRORS =    0
```

DESIGN PROBLEM :

TRAFFIC LIGHTS

You are to design and microcode a controller (using the Am2910!) to handle the following intersection:



There are four lights for straight-through traffic:

RED	YELLOW	GREEN	
NA	15s	80s	MAIN STREET
NA	15s	40s	SIDE STREET

There are four lights for protected left turn traffic:

RED	YELLOW	GREEN ARROW	
NA	10s	40s	BOTH STREETS

The four sensors, A, B, C, D, produce the SLT and MLT signals (side left turn, main left turn).

In case you are wondering, Sunnyvale really has a light that works like this! Precocious students have been known to find it.

To make the problem interesting, there are a few added considerations. If there is an accident, there is a manual override which allows all of the lights to be set to RED FLASH. And, because day traffic is heavier than night traffic, the controller can sense a control signal (timer generated if you wish) that tells it if it is day (normal operation), or night, when the straight-through lights on main street are set to YELLOW FLASH and all others are set to RED FLASH. Remember, you must be able to go back and forth:

```
              DAY <----> NIGHT
              DAY <----> MANUAL EMERGENCY
            NIGHT <----> MANUAL EMERGENCY
```

AND

```
      DAY ----> MANUAL EMERGENCY ----> NIGHT
    NIGHT ----> MANUAL EMERGENCY ----> DAY
```

If you have added up all the different lights, there are 5 different states: RED, YELLOW, GREEN, RED FLASH, YELLOW FLASH. We will assume that GREEN and GREEN ARROW are the same.

Five states means three lines of encoded controls into the individual traffic lights:

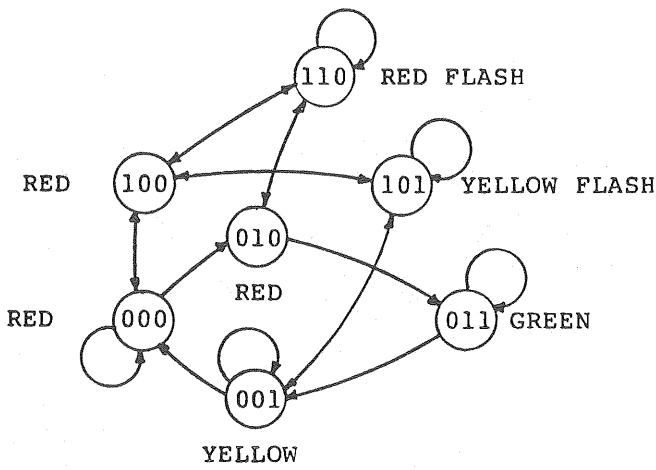
$$I_2 I_1 I_0$$

To make the problem even more interesting, there is a constraint on the way that the lights may be sequenced (remember that you are learning how to implement under constraints - this is a given constraint). Basically, the sequence must be a grey code, ie., only one signal line may change per clock step per light.

YOU HAVE A 5 SECOND CLOCK

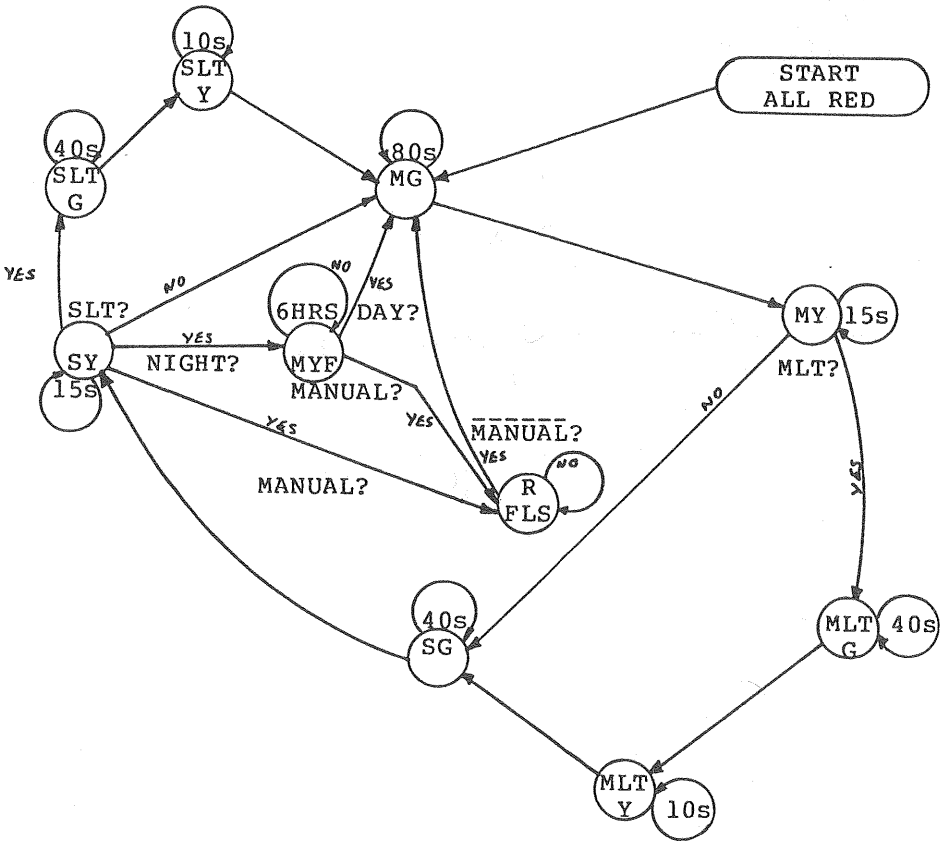
SAMPLE SEQUENCE

I_2	I_1	I_0	
0	0	0	RED
0	1	0	RED
1	0	0	RED
0	0	1	YELLOW
0	1	1	GREEN
1	0	1	YELLOW FLASH
1	1	0	RED FLASH

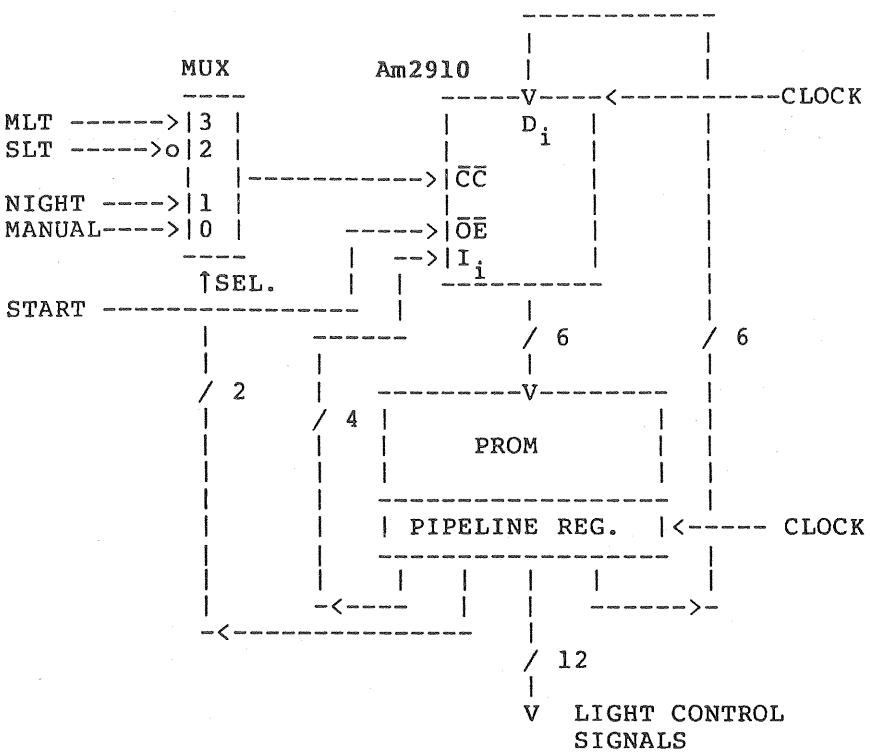


This is an example and happens to be what the code that is shown in the next pages was written to support. You can modify this slightly and reduce your code (the record so far is 27 microinstructions).

This is the stable-state diagram (does not show transitions). There is a subtle error or two here - you find them!



BASIC CONTROLLER HARDWARE



The output enable of the pipeline register is grounded. The output enables generated by the Am2910 are unused.

THE MICROPROGRAM (Draft version)

		SEQUENCE CONTROL			LIGHT CONTROL				
DEC ADR	LABEL	2910 INSTR	MUX TEST	COUNTER or BRANCH	S	MLT	SLT	M	COMMENT
0	MAIN	LDCT		15	R0	R0	R0	R2	LOAD CNTR
1	MLP	RPCT		MLP	R0	R0	R0	G	LOOP PL-MAIN
2		CONT			R0	R0	R0	Y	
3		CJP	MLT ?	MLT	R0	R0	R0	Y	MAIN LT TEST
4		LDCT		7	R2	R0	R0	Y	
5	SIDE	RPCT		SIDE	G	R0	R0	R0	LOOP PL- SIDE
6		CJP	MANUAL?	MANUAL	Y	R0	R0	R0	MANUAL OVERRIDE
7		CJP	NIGHT?	NIGHT	Y	R0	R0	R0	NIGHT TEST?
8		CJP	$\overline{SLT}$?	MAIN	Y	R0	R0	R0	SIDE LT TEST
9	SLT	LDCT		7	R0	R0	R2	R0	SIDE LT LOOP
10	SLTLP	RPCT		SLTLP	R0	R0	G	R0	LOOP PL
11		CONT			R0	R0	Y	R0	
12		JMAP		MAIN	R0	R0	Y	R0	MAP TIED TO PL
13	MLT	LDCT		7	R0	R2	R0	Y	MAIN LT LOOP
14	MLTLP	RPCT		MLTLP	R0	G	R0	R0	LOOP PL
15		LDCT		7	R0	Y	R0	R0	
16		JMAP		SIDE	R2	Y	R0	R0	
17	NIGHT	CONT			Y	R0	R0	R0	NIGHT LOOP
18		CONT			R0	R0	R0	R4	
19		LDCT		16	R4	R4	R4	YF	
20	NITLP	RPCT		NITLP	RF	RF	RF	YF	LOOP PL
21		LDCT		16	RF	RF	RF	YF	
22		CJP	MANUAL?	FIX	RF	RF	RF	YF	MANUAL OVERRIDE?
23		CJP	NIGHT?	NITLP	RF	RF	RF	YF	CONTINUE NIGHT?
24		CONT			R4	R4	R4	R4	
25		JMAP		MAIN	R0	R0	R0	R0	
26	MANUAL	CONT			Y	R0	R0	R0	MANUAL
27		CONT			Y	R0	R0	R0	
28		CONT			R0	R0	R0	R0	
29		LDCT		16	R4	R4	R4	R4	
30	ENTRY	RPCT		ENTRY	RF	RF	RF	RF	LOOP PL
31		LDCT		16	RF	RF	RF	RF	
32		CJP	MANUAL?	ENTRY	RF	RF	RF	RF	CONTINUE MANUAL?
33		CONT			R4	R4	R4	R4	
34		JMAP		MAIN	R0	R0	R0	R0	
35	FIX	JMAP		ENTRY	RF	RF	RF	R4	ADJUST SEQ FOR MANUAL
•									
•									EMPTY AREA
•									
63		JZ		MAIN	R0	R0	R0	R0	JUMP ZERO AND RESET

Review the preceeding microcode carefully and note the following.

a. The actual numerical address is given, in this case in decimal but HEX would have been even better. Whichever you use, note that you label it.

b. Labels are used and they have some relevancy to what is happening in the program. MLT refers to main left turn; NITLP is the RPCT loop for nighttime operation; SIDE is the normal, straight traffic, side street operation, etc.

c. All sequence control (microprogram sequence) fields are grouped together and characterize the individual microinstructions. The Am2910 instruction is given first, followed by the conditional MUX select field (to select which signal line is to be tested), and then the branch address field.

d. The branch address field is an overlay field whose meaning at any particular time is controlled by the Am2910 instruction. This is called "bit steering". The values for the cases when this field is a branch address field are labels. The values for the cases when this field is a counter field are given a decimal numerics. If this field were also a status-mask field for the Am2914, which is possible, those values might be given as HEX values or even better in mnemonics unique to the usage. The point is, when a field serves more than one purpose, make sure that the code is clear as to the usage in any given microinstruction.

e. The controls for the traffic lights are grouped and it is assumed that each field controls two lights. The mnemonics go with the light sequence diagram (R0 = 000 which is RED, R2 = 010 which is also RED, RF and YF are the flashing lights, etc.).

f. Note the comment field. This is important! Comments help someone else read your program and they help you to remember the program when updates, enhancements are being made.

g. Also make note of the vertical and horizontal lines that have been drawn throughout the microprogram. The horizontal lines group the various operations together and each group has a meaningful label. The various vertical lines delineate each field but also group the fields into the microprogram sequence control and light control functions. The vertical lines are not hard to add into the microprogram as you would write it on a AMC SYS/29 (use TAB to allign the fields). The horizontal lines can be easily achieved by using a comment-only line (a semicolon and *s). Whatever you use, the point is to visually separate the code into the different functions for the benefit of the humans who must read it.

h. Examine the testing that occurs following the side street green cycles. There are three tests being made: 1) is there an emergency condition (MANUAL?); 2) is it time to switch to night operation (NIGHT?); and 3) is a protected left turn requested (SLT?). These three tests are being done in the proper priority order, i.e., most important tested first. This is a good example of "polled interrupt" where the microprogram must test for each condition, one at a time.

As an exercise, correct the "errors" that you detect but also reduce the microprogram to fit into three 32x8 PROMS.

LAB TWO:

TRAFFIC LIGHT DEF FILE

LIBRARY

THE AmSYS/29 SYSTEM DISK CONTAINS A FILE CALLED "AM2900.LIB"

THIS FILE CONTAINS DEFINITIONS FOR MOST OF THE Am2900 PARTS, SOME AS COMMENT STATEMENTS, AND DUPLICATION OF MNEMONICS IS PRESENT

YOU ALSO HAVE AM2903.DEF ON YOUR DATA DISK WHICH IS A SUITABLE MASTER FILE FOR AM2903, AM2910, AND A PARTIAL REFERENCE FOR AM2904

BY EDITING THIS FILE A PARTIAL DEFINITION FILE CAN EXIST WITH VERY LITTLE EFFORT - YOU HAVE ALREADY USED THIS TO CREATE PART OF B:LIGHT.DEF (IN LAB ONE)

- POWER UP THE SYSTEMS

- DISPLAY B:LIGHT.DEF

- PRINT THE FILE FOR REFERENCE

```
A>Pc
```

ASSIGNMENT

USING THE COFFEE.DEF AND THE AM2903.DEF FILES AS REFERENCES YOU ARE TO CREATE THE .DEF FILE FOR THE ADVANCED TRAFFIC LIGHT DESIGN PROBLEM (DISCUSSED IN ED2900A)

- WORK IN TEAMS

- WRITE OUT A DRAFT FILE BEFORE TRYING TO INPUT

- TARGET IS SOME MINIMUM FILE WHICH WILL ALLOW YOU TO WRITE AT LEAST 5 LINES OF MICROCODE

LAB TWO

- POWER ON ALL UNITS
- SIGN ON TO THE SYSTEM
- USING EDITOR, CREATE THE DEFINITION FILE FOR THE TRAFFIC LIGHT ON THE B DRIVE
- USE THE FILENAME "B:LIGHT.DEF"
- EXIT THE EDITOR WHEN YOU ARE SATISFIED

- ASSEMBLE THE DEFINITION FILE BY TYPING

```
B:(Ret)
AMDASM P1 LIGHT
```

- IF YOU HAVE ERRORS, TYPE:

```
TYPE B:LIGHT.P1L
        USE Sc TO HALT SCREEN
        USE Pc TO LIST ON PRINTER
```

OR

```
DISPL B:LIGHT.P1L
        USE Ret TO PAGE FILE
        USE E Ret TO EXIT DISPL PROGRAM
```

- DEBUG USING THE EDITOR

PURPOSE: FAMILIARIZATION WITH THE USE OF THE EDITOR; EXPERIENCE WITH A DEF FILE.

CREATING A NEW FILE

TYPE --> ED B:LIGHT.DEF

THE SYSTEM WILL COME BACK WITH "NEW FILE" REPLY

ENTER YOUR TEXT (ALWAYS DEVELOP YOUR PROBLEM ON PAPER USING STRUCTURED TECHNIQUES, FLOWCHART, TABLES, ETC.)

WHEN FINISHED,

TYPE --> Zc

B#T

EXAMINE THE FILE FOR ANY ERRORS

IF ERRORS, CORRECT THEM

WHEN COMPLETED,

TYPE --> E

YOU ARE BACK TO THE SYSTEM

A>

EDITING "LIGHT.DEF" ON THE B DRIVE WHILE SIGNED ON TO THE A DRIVE (A> IS PROMPT):

TYPE --> ED B:LIGHT.DEF

THE SYSTEM WILL RESPOND WITH AN ASTERISK AS THE EDITOR PROMPT:

*

(ALL OF THE FOLLOWING ASSUME A CARRIAGE RETURN)

TYPE --> #A

THIS WILL READ THE FILE INTO THE WORKSPACE (MEMORY BUFFER)

TYPE --> B#T

THIS DISPLAYS THE ENTIRE FILE ON THE SCREEN

TO CHANGE OR ALTER A LINE THE CP (CHARACTER POINTER) HAS TO BE MOVED TO THE END OF THE LINE

TYPE --> F<line or string to be changed>Zc0LT

0LT (ZERO L T)

TYPE --> KI

THIS KILLS THE OLD LINE AND REQUESTS AN INSERT

NOW TYPE --> <insert the new data>
Zc

(note: <...> means you fill in - DO NOT TYPE THE "<" or ">"!)

THE SYSTEM WILL COME BACK WITH THE "*" PROMPT

TO CHANGE ANY OTHER ERRORS REPEAT THE INSTRUCTION

- ALWAYS GO TO THE TOP OF THE FILE BY TYPING "B" BEFORE SEARCHING FOR A STRING OR A LINE TO BE CHANGED (PROOFING YOUR FILE NEVER HURTS)

- THE "F" (SEARCH) COMMAND ALWAYS GOES FORWARD FROM THE CP POSITION

- AFTER ALL OF THE ERRORS ARE CORRECTED

TYPE --> E

THIS WILL CAUSE YOU TO EXIT EDITOR AND RETURN TO THE SYSTEM YOU WILL SEE:

A>

TYPE --> B:(Ret)

TYPE --> AMDASM P1 LIGHT

IF THERE ARE ANY ERRORS PRINT OUT A COPY OF THE LISTING AS FOLLOWS:

TURN ON PRINTER

ENABLE PRINTER BY TYPING Pc [IF NOT ENABLED]

TYPE --> TYPE B:LIGHT.PlL

(TYPE IS A VALID COMMAND)

GET BACK TO THE EDITOR AND CORRECT THE ERRORS

REASSEMBLE

LAB THREE:

TRAFFIC LIGHT SRC FILE

LAB THREE - SRC FILE

- USING THE EDITOR AS YOU DID FOR LAB TWO
    - CREATE THE FILE
      B:LIGHT.SRC
    - REFER TO YOUR DEF FILE
      B:LIGHT.DEF
- WHEN YOU HAVE FINISHED CREATION
    - PROOF THE FILE BY PRINTING IT OUT AND READING IT
    - CROSS CHECK YOUR MNEMONICS WITH THE DEF FILE
      COMMON ERROR IS MISSPELLING
- ASSEMBLE THE SRC FILE VIA:

```
B>AMDASM P2 LIGHT
```

- CORRECT ERRORS, IF ANY
    - LIST FILE VIA

```
A>TYPE B:LIGHT.P2L
        Pc SHOULD BE ENABLED
```

    - DISPLAY FILE VIA:

```
A>DISPL B:LIGHT.P2L
```

- REASSEMBLE UNTIL CORRECT

- - - YOU NEED AT LEAST FIVE GOOD MICROINSTRUCTIONS - - -

.SRC DESIGN PROCEDURE

- WRITE OUT THE MICROINSTRUCTIONS BEFORE SIGNING ONTO THE SYSTEM
- HAVE A COPY OF THE DEF FILE HANDY FOR REFERENCE

  DON'T GUESS AT THE CORRECT MNEMONIC
- USE COMMENTS, LOTS OF THEM!
- DEF STATEMENTS CAN BE ADDED TO DEF FILE IF YOU NEED THEM

  (RATHER THAN FREE FORMAT IN .SRC)
- IN ADDITION TO THE DOCUMENTATION IN THE DEF FILE, ADD FORMAT DOCUMENTATION TO THE SRC FILE

  THIS ALLOWS READER TO CHECK WHAT THE FIELD IS SUPPOSED TO BE DOING, ALLOWS CHECK ON MISALIGNMENT

  (CHECK OUT THE K-1 KIT FILE IN THE MANUAL

  CAN YOU MAKE SENSE OUT OF IT?)

EXAMPLE

```
;
;  2910    COND   BR ADDR  2901    2901    2901    Cin  A     B    ...
;  INSTR   MUX    COUNTER  SOURCE  FUNC    DESTIN  SEL  ADDR  ADDR ...
;  (4)     (2)    (6)      (3)     (3)     (3)     (2)  (4)   (4)  ...
;  CONT    FAIL   X        X       X       NOP     LOW  R0    R0   ...
```

TRAFFIC LIGHT

```
TITLE TRAFFIC LIGHT DEMO FILE
;
WORD 24                                                    .DEF
;
; ------------------------
; AM2910 INSTRUCTION SET
; ------------------------
;
; FORMED BY LAB ONE PIP EXERCISE
;
;
JMAP:           EQU     H#2             ; UNCOND JUMP TO MEMORY MAP (Di)
CJP:            EQU     H#3             ; COND JUMP PIPELINE
PUSH:           EQU     H#4             ; PUSH STACK, LOAD REG MAYBE, CONT
JSRP:           EQU     H#5             ; JUMP SUB FROM REG (F) OR PIPE(T)
CJV:            EQU     H#6             ; COND JUMP TO VECTOR INTER (Di)
JRP:            EQU     H#7             ; JUMP TO REG (F) OR PIPE (T)
RFCT:           EQU     H#8             ; DO LOOP REPEAT UNTIL CTR=0 - STACK
RPCT:           EQU     H#9             ; DO LOOP UNTIL CTR=0 - PIPE
CRTN:           EQU     H#A             ; COND RETURN, POP STACK (T)
CJPP:           EQU     H#B             ; COND JUMP PIPELINE, POP STACK
LDCT:           EQU     H#C             ; LOAD REGISTER, CONTINUE
LOOP:           EQU     H#D             ; DO LOOP UNTIL TEST=T - STACK
CONT:           EQU     H#E             ; CONTINUE
;
; TWB WILL BE INCOMPLETE - FIX IT
;
TWB:            EQU     H#F             ; THREE WAY BRANCH (DEAD MAN TIMER)
;
; -------------------------
; CONDITIONAL MUX SELECT
; -------------------------
;
MLT:            EQU     B#11            ; MAIN LEFT TURN REQUEST
NOSLT:          EQU     B#10            ; NO SIDE LEFT TURN REQUEST
NIGHT:          EQU     B#01            ; NIGHT OPERATION
MANUAL:         EQU     B#00            ; EMERGENCY REQUEST
;
; -------------------------
; LIGHT CONTROL SIGNALS
; (SEQUENCE STATES)
; -------------------------
;
R0:             EQU     Q#0
Y:              EQU     Q#1
R2:             EQU     Q#2
G:              EQU     Q#3
R4:             EQU     Q#4
YF:             EQU     Q#5
RF6:            EQU     Q#6
RF7:            EQU     Q#7
;
;
; ---------------------------------
; MICROWORD DEFINITION
; ---------------------------------
;
LITE:   DEF     4VH#E, 2VB#00, 6V$X, 3VQ#7, 3VQ#7, 3VQ#7, 3VQ#7
; DEFAULTS      CONT   MANUAL  X     RF7     RF7     RF7     RF7
;
END
```

PARTIAL .SRC

```
TITLE TRAFFIC LIGHT SOURCE FILE
;
; TEMPORARY EQUATES SO PARTIAL CODE WILL RUN - REMOVE WHEN
; STATEMENTS WITH THESE LABELS ARE ADDED!!!!!
;
EMER:           EQU     D#26
EVNG:           EQU     D#17
GOTO:           EQU     H#2                                     ; GOTO = JMAP
LMLT:           EQU     D#13
;
;
; MAIN STREET SEQUENCE
;-------------------------------------------------------------------------
MAIN:   LITE LDCT,  ,   H#F,    R0, R0, R0, R2          ; LOAD COUNTER
MLP:    LITE RPCT,  ,   MLP,    R0, R0, R0, G          ; MAIN GREEN LOOP
        LITE    ,   ,     ,    R0, R0, R0, Y
        LITE CJP, MLT,  LMLT,   R0, R0, R0, Y          ; NEED LEFT TURN?
        LITE LDCT,  ,   H#7,    R2, R0, R0, Y
;-------------------------------------------------------------------------
; SIDE STREET SEQUENCE
;-------------------------------------------------------------------------
SIDE:   LITE RPCT,  ,   SIDE,   G,  R0, R0, R0          ; SIDE GREEN LOOP
        LITE CJP,MANUAL,EMER,   Y,  R0, R0, R0          ; PRIORITY POLLING
        LITE CJP,NIGHT, EVNG,   Y,  R0, R0, R0          ;  SEQUENCE
        LITE CJP,NOSLT, MAIN,   Y,  R0, R0, R0
;-------------------------------------------------------------------------
; SIDE STREET LEFT TURN SEQUENCE
;-------------------------------------------------------------------------
LSLT:   LITE LDCT,  ,   H#7,    R0, R0, R2, R0          ;
SLTLP:  LITE RPCT,  ,  SLTLP,   R0, R0, G,  R0          ; SIDE LEFT LOOP
        LITE    ,   ,     ,   R0, R0, Y,  R0          ; TIME YELLOW LIGHT
        LITE GOTO,  ,   MAIN,   R0, R0, Y,  R0
;-------------------------------------------------------------------------
; ETC.
;
;
END
```

.P2L

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.4
TRAFFIC LIGHT SOURCE FILE

      ;
      ; TEMPORARY EQUATES SO PARTIAL CODE WILL RUN - REMOVE WHEN
      ; STATEMENTS WITH THESE LABELS ARE ADDED!!!!!
      ;
 001A EMER:          EQU     D#26
 0011 EVNG:          EQU     D#17
 0002 GOTO:          EQU     H#2                          ; GOTO = JMAP
      LMLT:          EQU     D#13
      ;
      ;
      ; MAIN STREET SEQUENCE
 000D ;-------------------------------------------------------------------
 0000 MAIN:   LITE LDCT,  ,   H#F,   R0, R0, R0, R2       ; LOAD COUNTER
 0001 MLP:    LITE RPCT,  ,   MLP,   R0, R0, R0, G        ; MAIN GREEN LOOP
 0002         LITE    ,   ,     ,   R0, R0, R0, Y
 0003         LITE CJP, MLT,  LMLT,  R0, R0, R0, Y        ; NEED LEFT TURN?
 0004         LITE LDCT,  ,   H#7,   R2, R0, R0, Y
      ;-------------------------------------------------------------------
      ; SIDE STREET SEQUENCE
      ;-------------------------------------------------------------------
 0005 SIDE:   LITE RPCT,  ,   SIDE,  G,  R0, R0, R0       ; SIDE GREEN LOOP
 0006         LITE CJP,MANUAL,EMER,  Y,  R0, R0, R0       ; PRIORITY POLLING
 0007         LITE CJP,NIGHT, EVNG,  Y,  R0, R0, R0       ;  SEQUENCE
 0008         LITE CJP,NOSLT, MAIN,  Y,  R0, R0, R0
      ;-------------------------------------------------------------------
      ; SIDE STREET LEFT TURN SEQUENCE
      ;-------------------------------------------------------------------
 0009 LSLT:   LITE LDCT,  ,   H#7,   R0, R0, R2, R0       ;
 000A SLTLP:  LITE RPCT,  ,  SLTLP,  R0, R0, G,  R0       ; SIDE LEFT LOOP
 000B         LITE    ,   ,     ,   R0, R0, Y,  R0       ; TIME YELLOW
 000C         LITE GOTO,  ,   MAIN,  R0, R0, Y,  R0
      ;-------------------------------------------------------------------
      ; ETC.
      ;
      ;
      END
```

CODE

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.4
TRAFFIC LIGHT SOURCE FILE

 0000 1100000011110000 00000010
 0001 1001000000010000 00000011
 0002 111000XXXXXX0000 00000001
 0003 0011110011010000 00000001
 0004 1100000001110100 00000001
 0005 1001000001010110 00000000
 0006 0011000110100010 00000000
 0007 0011010100010010 00000000
 0008 0011100000000010 00000000
 0009 1100000001110000 00010000
 000A 1001000010100000 00011000
 000B 111000XXXXXX0000 00001000
 000C 0010000000000000 00001000
```

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.4
TRAFFIC LIGHT SOURCE FILE

 SYMBOLS

  CJP       0003
  CJPP      000B
  CJV       0006
  CONT      000E
  CRTN      000A
  EMER      001A
  EVNG      0011
  G         0003
  GOTO      0002
  JMAP      0002
  JRP       0007
  JSRP      0005
  LDCT      000C
  LMLT      000D
  LOOP      000D
  LSLT      0009
  MAIN      0000
  MANUAL    0000
  MLP       0001
  MLT       0003
  NIGHT     0001
  NOSLT     0002
  PUSH      0004
  R0        0000
  R2        0002
  R4        0004
  RF6       0006
  RF7       0007
  RFCT      0008
  RPCT      0009
  SIDE      0005
  SLTLP     000A
  TWB       000F
  Y         0001
  YF        0005

 TOTAL PHASE 2 ERRORS =    0
```

SYMBOL TABLE

DESIGN PROBLEM:

A SIMPLE DATA MONITOR

PROBLEM:

Design a simple data-gatherer such that the data input is read into a 4-bit data-input register. Assume that the data is always ready to be read into the data-in register; we can add a ready-to-receive bit later for "handshaking". When data is output, the monitor waits for an ACK signal before proceeding with its operation. The monitor is to have a 12-bit RALU. A minimum of five registers are to behave as counters.

DEVICES:

Use an Am2910 and an ALU made up out of Am290ls.

DESIGN APPROACH:

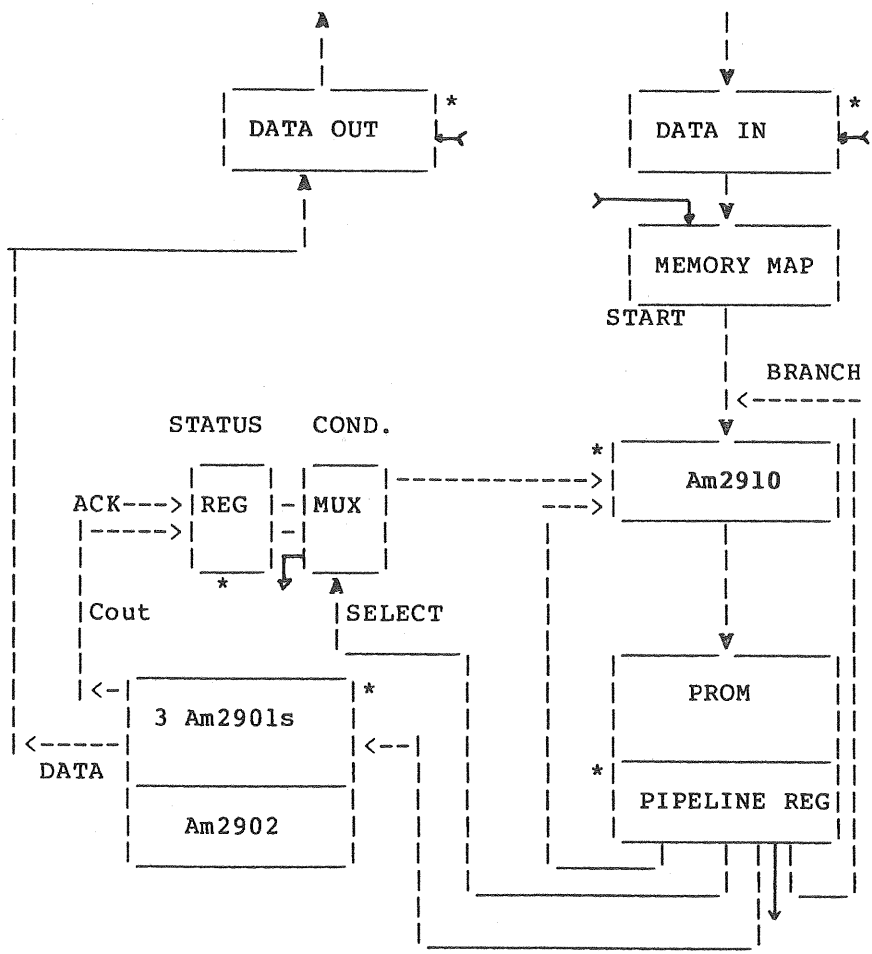
Use a pipelined PROM control memory, a status register (1-bit) and a memory map to decode the data input.

MICROPROGRAM DESCRIPTION:

```
START
    1. INITIALIZE REGISTERS
        R0 <-- 0    R1 <-- 0    R2 <-- 0    R3 <-- 0    R4 <-- 0
NEXT:2. LOAD DATA-IN REGISTER
         DATA-IN <-- DATA-BUS
    3. IF DATA-IN = 0
          THEN R0 <-- R0 + 1
               IF Cn+4 = 1
                  THEN JUMP SUB0
                       R1 --> DATA-OUT
          ELSE R1 <-- R1 + 1          ALL NUMBERS ARE POSITIVE
               IF Cn+4 = 1
                  THEN JUMP SUB0
                       R0 --> DATA-OUT
    4. CASE BRANCH
       IF 0 ≤ DATA-IN ≤ 5
           THEN R2 <-- R2 + 1
       IF 5 < DATA-IN ≤ 10
           THEN R3 <-- R3 + 1
       IF 10 < DATA-IN
           THEN R4 <-- R4 + 1
    5. BRANCH TO NEXT
```

```
SUB0:6.  R2 --> DATA-OUT
     7.  IF ĀC̄K̄
             THEN WAIT
     8.  R3 --> DATA-OUT
     9.  IF ĀC̄K̄
             THEN WAIT
     10. R4 --> DATA-OUT
     11. IF ĀC̄K̄
             THEN WAIT
     12. RESET REGISTERS
          R2 <-- 0     R3 <-- 0     R4 <-- 0
     13. RETURN
```

BASIC HARDWARE



* CLOCK INTO: Am2910
Am2901s
PIPELINE REGISTER
DATA REGISTERS (12 BITS AND 4 BITS)
STATUS REGISTER (2 BITS)

MICROWORD FORMAT [DRAFT VERSION]

LABEL ADDR	2910 INSTR	COND MUX	BR ADDR COUNTER	SRCE A	FUNC L	DEST U	CARRY IN	A ADDR	B ADDR	DATA IN CTRL	DATA OUT CTRL

The final number of bits that are required for the microword is a function of the final microprogram. An initial guess can be made by examining what exists so far in the microword format.

1. LABEL/ADDR — This will be the actual in-place address and is filled in last. Labels are filled in as created.
2. 2910 INSTR — The 4-bit instruction field for the microprogram controller
3. COND MUX — The conditional MUX select control is 1 bit so far.
4. BR ADDR/COUNTER — The branch address field 4-6 bits is a good guess so far.
5. ALU CONTROL — These three fields are each 3 bits.
6. CARRY IN — For the ALU, 1 bit.
7. A ADDR, B ADDR — Selects the ALU registers, need 5 registers, so two 3 bit fields.
8. DATA IN CTRL — Load the DATA-IN register, 1 bit.
9. DATA OUT CTRL — Load the DATA-OUT register, 1 bit.

Both 8 and 9 will expand to 2 bits if enables are also needed.

Total # bits: 28 bits ± 1

LABEL /ADDR	2910 INSTR	COND MUX	BR ADDR COUNTER	SRCE A	FUNC L	DEST U	C IN	ADDR A	ADDR B	DATA-IN CONTROL	DATA-OUT CONTROL	MEMORY MAP SEL
* FIRST WRITE CODE TO I N I T I A L I Z E REGISTERS												
START	CONT	#	#	AB	EXOR	RAMF	#	R0	R0	NOLOAD	NOLOAD	0
1	CONT	#	#	AB	EXOR	RAMF	#	R1	R1	NOLOAD	NOLOAD	0
2	CONT	#	#	AB	EXOR	RAMF	#	R2	R2	NOLOAD	NOLOAD	0
3	CONT	#	#	AB	EXOR	RAMF	#	R3	R3	NOLOAD	NOLOAD	0
4	CONT	#	#	AB	EXOR	RAMF	#	R4	R4	NOLOAD	NOLOAD	0
NEXT	CONT	#	#	#	#	NOP	#	#	#	LOAD	NOLOAD	0

NOW CONSIDER HOW YOU ARE GOING TO HANDLE THE MULTIPLE BRANCHES? ONE SOLUTION WHICH HAS BEEN STARTED HERE IS TO HAVE A LARGER MEMORY MAP AND AN EXTRA ADDRESS LINE INTO IT SO THAT THE INPUT DATA CAN BE DECODED TWICE USING DIFFERENT PORTIONS OF THE MAP FOR EACH ACCESS.

ANOTHER SOLUTION WOULD BE TO READ THE DATA INTO AN ALU REGISTER AND USE THE ZERO TEST STATUS TO MAKE THE FIRST DECISION AND USE THE MEMORY MAP FOR THE CASE BRANCH DECISION. THE MICROWORD WOULD BE THE SAME WIDTH IN BOTH CASES (THE Z STATUS REQUIRES A WIDER STATUS REGISTER AND CONDITION CODE MUX).

WE WILL EXAMINE BOTH SOLUTIONS.

THE DOUBLE MAP:

LABEL /ADDR	2910 INSTR	COND MUX	BR ADDR COUNTER	SRCE A	FUNC L	DEST U	C IN	ADDR A	ADDR B	DATA-IN CONTROL	DATA-OUT CONTROL	MEMORY MAP SEL
6	JMAP	#	#	#	#	NOP	#	#	#	NOLOAD	NOLOAD	0
DISZ	CONT	#	#	AZ	ADD	RAMF	H	R0	R0	NOLOAD	NOLOAD	0
8	JSUB	Cn+4	SUB0	#	#	NOP	#	#	#	NOLOAD	NOLOAD	0
9	CONT	#	#	AB	OR	RAMA	#	R1	R1	NOLOAD	LOAD	0
WAITA	CJP	NOACK	WAITA	#	#	NOP	#	#	#	NOLOAD	NOLOAD	0
B	CJP	PASS	CASE	#	#	NOP	#	#	#	NOLOAD	NOLOAD	1
DGRZ	CONT	#	#	AZ	ADD	RAMF	H	R1	R1	NOLOAD	NOLOAD	0
D	JSUB	Cn+4	SUB0	#	#	NOP	#	#	#	NOLOAD	NOLOAD	0
E	CONT	#	#	AB	OR	RAMA	#	R0	R0	NOLOAD	LOAD	0
WAITB	CJP	NOACK	WAITB	#	#	NOP	#	#	#	NOLOAD	NOLOAD	1
CASE	JMAP	#	#	#	#	NOP	#	#	#	NOLOAD	NOLOAD	1
L5	CJP	PASS	NEXT	AZ	ADD	RAMF	H	R2	R2	NOLOAD	NOLOAD	0
GR5L10	CJP	PASS	NEXT	AZ	ADD	RAMF	H	R3	R3	NOLOAD	NOLOAD	0
GR10	CJP	PASS	NEXT	AZ	ADD	RAMF	H	R4	R4	NOLOAD	NOLOAD	0
*SUBROUTINE												
SUB0	CONT	#	#	AB	OR	RAMA	#	R2	R2	NOLOAD	LOAD	0
WAIT1	CJP	NOACK	WAIT1	#	#	NOP	#	#	#	NOLOAD	NOLOAD	0
16	CONT	#	#	AB	OR	RAMA	#	R3	R3	NOLOAD	LOAD	0
WAIT2	CJP	NOACK	WAIT2	#	#	NOP	#	#	#	NOLOAD	NOLOAD	0
18	CONT	#	#	AB	OR	RAMA	#	R4	R4	NOLOAD	LOAD	0
WAIT3	CJP	NOACK	WAIT3	#	#	NOP	#	#	#	NOLOAD	NOLOAD	0
1A	CONT	#	#	AB	EXOR	RAMF	#	R2	R2	NOLOAD	NOLOAD	0
1B	CONT	#	#	AB	EXOR	RAMF	#	R3	R3	NOLOAD	NOLOAD	0
1C	CONT	#	#	AB	EXOR	RAMF	#	R4	R4	NOLOAD	NOLOAD	0
1D	CRTN	PASS	#	#	#	NOP	#	#	#	NOLOAD	NOLOAD	0

TOTAL NUMBER OF MICROWORDS = 1D (HEX) = 29

TOTAL NUMBER OF BRANCH ADDRESS LINES = 5

AM2900.LIB FILE

(1976-7)

```
; TITLE        T H E   A M 2 9 0 0   F A M I L Y   M N E M O N I C S
;       ****************************************************
;
;
WORD 1          ; INSERTED AS DUMMY TO PROCESS DEFINITIONS
;13 DECEMBER 1976 JRM
;UPDATED SEPT 28, 1977
;
;AM2901 INSTRUCTION SET
;
;REGISTER DEFINITIONS
;
R0:     EQU     H#0
R1:     EQU     H#1
R2:     EQU     H#2
R3:     EQU     H#3
R4:     EQU     H#4
R5:     EQU     H#5
R6:     EQU     H#6
R7:     EQU     H#7
R8:     EQU     H#8
R9:     EQU     H#9
R10:    EQU     H#A
R11:    EQU     H#B
R12:    EQU     H#C
R13:    EQU     H#D
R14:    EQU     H#E
R15:    EQU     H#F
;
;AM2901 SOURCE OPERANDS (R S)
;
AQ:     EQU     Q#0
AB:     EQU     Q#1
ZQ:     EQU     Q#2
ZB:     EQU     Q#3
ZA:     EQU     Q#4
DA:     EQU     Q#5
DQ:     EQU     Q#6
DZ:     EQU     Q#7
;
;AM2901 ALU FUNCTIONS (R FUNCTION S)
;
ADD:    EQU     Q#0
SUBR:   EQU     Q#1
SUBS:   EQU     Q#2
OR:     EQU     Q#3
AND:    EQU     Q#4
NOTRS:  EQU     Q#5
EXOR:   EQU     Q#6
EXNOR:  EQU     Q#7
;
;AM2901 DESTINATION CONTROL
;
QREG:   EQU     Q#0
NOP:    EQU     Q#1
```

```
RAMA:    EQU     Q#2
RAMF:    EQU     Q#3
RAMQD:   EQU     Q#4
RAMD:    EQU     Q#5
RAMQU:   EQU     Q#6
RAMU:    EQU     Q#7
;
;AM29811 INSTRUCTION SET
;
JZ:      EQU     H#0      ;JUMP TO ADDRESS ZERO
CJS:     EQU     H#1      ;CONDITIONAL JUMP TO SUBROUTINE WITH JUMP
                          ;ADDRESS IN THE PIPELINE REGISTER
JMAP:    EQU     H#2      ;JUMP TO ADDRESS AT MAPPING PROM OUTPUT
CJP:     EQU     H#3      ;CONDITIONAL JUMP TO ADDRESS IN PIPELINE
                          ;REGISTER
PUSH:    EQU     H#4      ;PUSH STACK AND CONDITIONALLY LOAD COUNTER
JSRP:    EQU     H#5      ;JUMP TO SUBROUTINE WITH STARTING ADDRESS
                          ;CONDITIONALLY SELECTED FROM THE AM2911
                          ;R-REGISTER OR PIPELINE ADDRESS
CJV:     EQU     H#6      ;CONDITIONAL JUMP TO VECTOR ADDRESS
JRP:     EQU     H#7      ;JUMP TO ADDRESS CONDITIONALLY SELECTED FROM
                          ;AM2911 R-REGISTER OR PIPELINE REGISTER
RFCT:    EQU     H#8      ;REPEAT LOOP IF COUNTER IS NOT EQUAL TO ZERO
RPCT:    EQU     H#9      ;REPEAT PIPELINE ADDRESS IF COUNTER IS NOT
                          ;EQUAL TO ZERO
CRTN:    EQU     H#A      ;CONDITIONAL RETURN FROM SUBROUTINE
CJPP:    EQU     H#B      ;CONDITIONAL JUMP TO PIPELINE ADDRESS AND POP
                          ;STACK
LDCT:    EQU     H#C      ;LOAD COUNTER AND CONTINUE
LOOP:    EQU     H#D      ;TEST END OF LOOP
CONT:    EQU     H#E      ;CONTINUE TO NEXT ADDRESS
JP:      EQU     H#F      ;JUMP TO PIPELINE REGISTER ADDRESS
;
;AM2910 MICROPROGRAM SEQUENCER
;
;DELETE ";" FROM AM2910 NMEMONICS IF NEEDED
;(DELETE IN EDITOR USING D-RETURN-RETURN)
;
;JZ:      EQU     H#0      ;JUMP ZERO (RESET)
;CJS:     EQU     H#1      ;CONDITIONAL JUMP SUBROUTINE PIPELINE
;JMAP:    EQU     H#2      ;JUMP MAP
;CJP:     EQU     H#3      ;CONDITIONAL JUMP PIPELINE
;PUSH:    EQU     H#4      ;PUSH/CONDITIONAL LOAD COUNTER
;JSRP:    EQU     H#5      ;CONDITIONAL JUMP SUBROUTINE R OR PIPELINE
;CJV:     EQU     H#6      ;CONDITIONAL JUMP VECTOR
;JRP:     EQU     H#7      ;CONDITIONAL JUMP R OR PIPELINE
;RFCT:    EQU     H#8      ;REPEAT LOOP, COUNTER NOT ZERO
;RPCT:    EQU     H#9      ;REPEAT PIPELINE, COUNTER NOT ZERO
;CRTN:    EQU     H#A      ;CONDITIONAL RETURN
;CJPP:    EQU     H#B      ;CONDITIONAL JUMP PIPELINE AND POP
;LDCT:    EQU     H#C      ;LOAD COUNTER AND CONTINUE
;LOOP:    EQU     H#D      ;TEST END LOOP
;CONT:    EQU     H#E      ;CONTINUE
;TWB:     EQU     H#F      ;THREE-WAY BRANCH
                          ;THREE-WAY DEFINITION
```

```
                ;PASS TEST - CONTINUE PC AND POP
                ;FAIL TEST - REPEAT LOOP IF COUNTER NOT ZERO
                ;FAIL TEST - JUMP PIPELINE AND POP IF COUNTER ZERO
;
;AM2914 INSTRUCTION SET
;
MCLR:   EQU     H#0     ;MASTER CLEAR
CLRIN:  EQU     H#1     ;CLEAR ALL INTERRUPTS
CLRMB:  EQU     H#2     ;CLEAR INTERRUPTS FROM M-BUS
CLRMR:  EQU     H#3     ;CLEAR INTERRUPTS FROM MASK REGISTER
CLRVC:  EQU     H#4     ;CLEAR INTERRUPT FROM LAST VECTOR READ
RDVC:   EQU     H#5     ;READ VECTOR
RDSTA:  EQU     H#6     ;READ STATUS REGISTER
RDM:    EQU     H#7     ;READ MASK REGISTER
SETM:   EQU     H#8     ;SET MASK REGISTER
LDSTA:  EQU     H#9     ;LOAD STATUS REGISTER
BCLRM:  EQU     H#A     ;BIT CLEAR MASK REGISTER
BSETM:  EQU     H#B     ;BIT SET MASK REGISTER
CLRM:   EQU     H#C     ;CLEAR MASK REGISTER
DISIN:  EQU     H#D     ;DISABLE INTERRUPT REQUEST
LDM:    EQU     H#E     ;LOAD MASK REGISTER
ENIN:   EQU     H#F     ;ENABLE INTERRUPT REQUEST
;
;AM2930 PROGRAM CONTROL UNIT
;
;NON-CONDITIONAL INSTRUCTIONS
;
PRST:   EQU     5H#00:  ;RESET
FPC:    EQU     5H#01:  ;FETCH PC
FR:     EQU     5H#02:  ;FETCH R
FD:     EQU     5H#03:  ;FETCH D
FRD:    EQU     5H#04:  ;FETCH R PLUS D
FPD:    EQU     5H#05:  ;FETCH PC PLUS D
FPR:    EQU     5H#06:  ;FETCH PC PLUS R
FSD:    EQU     5H#07:  ;FETCH S PLUS D
FPLR:   EQU     5H#08:  ;FETBH PC, LOAD R
FRDR:   EQU     5H#09:  ;FETCH R PLUS D, LOAD R
PLDR:   EQU     5H#0A:  ;LOAD R
PSHP:   EQU     5H#0B:  ;PUSH PC
PSHD:   EQU     5H#0C:  ;PUSH D
POPS:   EQU     5H#0D:  ;POP S
POPP:   EQU     5H#0E:  ;POP PC
PHLD:   EQU     5H#0F:  ;HOLD
;
;CONDITIONAL INSTRUCTIONS - FAIL TEST, EXECUTE FPC
;
JMPR:   EQU     5H#10:  ;JUMP R
JMPD:   EQU     5H#11:  ;JUMP D
JMPZ:   EQU     5H#12:  ;JUMP ZERO
JPRD:   EQU     5H#13:  ;JUMP R PLUS D
JPPD:   EQU     5H#14:  ;JUMP PC PLUS D
JPPR:   EQU     5H#15:  ;JUMP PC PLUS R
JSBR:   EQU     5H#16:  ;JUMP SUBROUTINE R
JSBD:   EQU     5H#17:  ;JUMP SUBROUTINE D
JSBZ:   EQU     5H#18:  ;JUMP SUBROUTINE ZERO
```

```
JSRD:    EQU      5H#19:   ;JUMP SUBROUTINE R PLUS D
JSPD:    EQU      5H#1A:   ;JUMP SUBROUTINE PC PLUS D
JSPR:    EQU      5H#1B:   ;JUMP SUBROUTINE PC PLUS R
RTS:     EQU      5H#1C:   ;RETURN S
RTSD:    EQU      5H#1D:   ;RETURN S PLUS D
CHLD:    EQU      5H#1E:   ;HOLD
PSUS:    EQU      5H#1F:   ;SUSPEND
;
;
;AM2932 PROGRAM CONTROL UNIT
;
;DELETE ";" FROM AM2932 NMEMONICS IF NEEDED
;
;PRST:   EQU      H#0      ;RESET
;PSUS:   EQU      H#1      ;SUSPEND
;PSHD:   EQU      H#2      ;PUSH D
;POPS:   EQU      H#3      ;POP STACK
;FPC:    EQU      H#4      ;FETCH PC
;JMPD:   EQU      H#5      ;JUMP D
;PSHP:   EQU      H#6      ;PUSH PC
;RTS:    EQU      H#7      ;RETURN STACK
;FR:     EQU      H#8      ;FETCH R
;FPR:    EQU      H#9      ;FETCH PC PLUS R
;FPLR:   EQU      H#A      ;FETCH PC, LOAD R
;JMPR:   EQU      H#B      ;JUMP R
;JPPR:   EQU      H#C      ;JUMP PC PLUS R
;JSBR:   EQU      H#D      ;JUMP SUBROUTINE R
;JSPR:   EQU      H#E      ;JUMP SUBROUTINE PC PLUS R
;PLDR:   EQU      H#F      ;LOAD R
;
;AM2940 DMA CONTROL UNIT
;
;INSTRUCTIONS
;
WRCR:    EQU      Q#0      ;WRITE CONTROL REGISTER
RDCR:    EQU      Q#1      ;READ CONTROL REGISTER
RDWC:    EQU      Q#2      ;READ WORD COUNTER
RDAC:    EQU      Q#3      ;READ ADDRESS COUNTER
REIN:    EQU      Q#4      ;REINITIALIZE COUNTERS
LDAD:    EQU      Q#5      ;LOAD ADDRESS
LDWC:    EQU      Q#6      ;LOAD WORD COUNT
ENCT:    EQU      Q#7      ;ENABLE COUNTERS
;
;CONTROL MODE BYTE
;NOTE - BITS 3 THROUGH 7 ARE DON'T CARE
;
WC1I:    EQU      8Q#0%    ;WORD COUNT EQUALS ONE, INCREMENT ADDRESS COUNTER
WCCI:    EQU      8Q#1%    ;WORD COUNT COMPARE, INCREMENT ADDRESS COUNTER
ADCI:    EQU      8Q#2%    ;ADDRESS COMPARE, INCREMENT ADDRESS COUNTER
WCOI:    EQU      8Q#3%    ;WORD COUNTER CARRY OUT, INCREMENT ADDRESS COUNTER
WC1D:    EQU      8Q#4%    ;WORD COUNT EQUALS ONE, DECREMENT ADDRESS COUNTER
WCCD:    EQU      8Q#5%    ;WORD COUNT COMPARE, DECREMENT ADDRESS COUNTER
ADCD:    EQU      8Q#6%    ;ADDRESS COMPARE, DECREMENT ADDRESS COUNTER
WCOD:    EQU      8Q#7%    ;WORD COUNTER CARRY OUT, DECREMENT ADDRESS COUNTER
;
```

```
; AM2903 INSTRUCTION SET  (7TH NOVEMBER, 1978    TD)
;
; ALU OPERAND SOURCES
;
; ALU SOURCE OPERAND R  (EA*)
;
RAMA:    EQU     B#0      ; R INPUT FROM RAM PORT A
DA:      EQU     B#1      ; R INPUT FROM DA BUS
;
; ALU SOURCE OPERAND S  (I0,OEB*)
;
RAMB:    EQU     B#00     ; S INPUT FROM RAM PORT B
DB:      EQU     B#01     ; S INPUT FROM DB BUS
Q:       EQU     B#10     ; IF (I4-I1)=0 THEN ALU OUTPUT F IS FORCED HIGH
                          ;    REGARDLESS OF INPUT,
                          ; ELSE S INPUT FROM Q REGISTER
;
; ALU FUNCTIONS  (I4,I3,I2,I1)
;
SPF:     EQU     H#0      ; SPECIAL FUNCTIONS, S OPERAND RAMB OR DB, BUT NOT Q
HIGH:    EQU     H#0      ; ALU OUTPUT F FORCED HIGH, S OPERAND MUST SPECIFY Q
SUBR:    EQU     H#1      ; S MINUS R
SUBS:    EQU     H#2      ; R MINUS S
ADD:     EQU     H#3      ; R ADD S
PASSS:   EQU     H#4      ; PASS S
COMPLS:  EQU     H#5      ; 2'S COMPLEMENT S
PASSR:   EQU     H#6      ; PASS R
COMPLR:  EQU     H#7      ; 2'S COMPLEMENT R
LOW:     EQU     H#8      ; ALU OUTPUT FORCED LOW
NOTRS:   EQU     H#9      ; COMPLEMENT R, AND WITH S
EXNOR:   EQU     H#A      ; R EXCLUSIVE NOR WITH S
EXOR:    EQU     H#B      ; R EXOR S
AND:     EQU     H#C      ; R AND S
NOR:     EQU     H#D      ; R NOR S
NAND:    EQU     H#E      ; R NAND S
OR:      EQU     H#F      ; R OR S
;
; SPECIAL FUNCTIONS  (I8,I7,I6,I5)
; I8-I5 SPECIFIES SPECIAL FUNCTIONS IF ALU=SPF, AND S OPERAND IS RAMB OR DB
;    OTHERWISE IT SPECIFIES ALU DESTINATION CONTROL
;
USMUL:   EQU     H#0      ; UNSIGNED MULTIPLY
TCMUL:   EQU     H#2      ; 2'S COMPLEMENT MULTIPLY
INC:     EQU     H#4      ; INCREMENT BY ONE OR TWO
SMTC:    EQU     H#5      ; SIGN MAGNITUDE - 2'S COMPLEMENT
TCMLS:   EQU     H#6      ; 2'S COMPLEMENT MULTIPLY LAST STEP
SLN:     EQU     H#8      ; SINGLE LENGTH NORMALIZE
DLN:     EQU     H#A      ; DOUBLE LENGTH NORMALIZE
TCDIV:   EQU     H#C      ; 2'S COMPLEMENT DIVISION
TCDC:    EQU     H#E      ; 2'S COMPLEMENT DIVISION CORRECTION
;
; ALU DESTINATION CONTROL  (I8,I7,I6,I5)
; I8-I5 SPECIFIES ALU DESTINATION CONTROL ONLY IF ALU IS NOT SPF
```

```
;
ADR:    EQU     H#0     ; ARITH SHIFT DOWN, RESULT INTO RAM
LDR:    EQU     H#1     ; LOGICAL SHIFT DOWN, RESULT INTO RAM
ADRQ:   EQU     H#2     ; ARITH SHIFT DOWN, RESULT INTO RAM AND Q
LDRQ:   EQU     H#3     ; LOGICAL SHIFT DOWN, RESULT INTO RAM AND Q
RPT:    EQU     H#4     ; RESULT INTO RAM, GENERATE PARITY
LDQP:   EQU     H#5     ; LOGICAL SHIFT DOWN Q, GENERATE PARITY
QPT:    EQU     H#6     ; RESULT INTO Q, GENERATE PARITY
RQPT:   EQU     H#7     ; RESULT INTO RAM AND Q, GENERATE PARITY
AUR:    EQU     H#8     ; ARITH SHIFT UP, RESULT INTO RAM
LUR:    EQU     H#9     ; LOGICAL SHIFT UP, RESULT INTO RAM
AURQ:   EQU     H#A     ; ARITH SHIFT UP, RESULT INTO RAM AND Q
LURQ:   EQU     H#B     ; LOGICAL SHIFT UP, RESULT INTO RAM AND Q
YBUS:   EQU     H#C     ; RESULT TO Y BUS ONLY
LUQ:    EQU     H#D     ; LOGICAL SHIFT UP Q
SEX:    EQU     H#E     ; SIGN EXTEND
RSEX:   EQU     H#F     ; RESULT TO RAM, SIGN EXTEND
;
        END
```

MASTER .DEF FILE

Am2903-2904-2910

(1980-1)

MASTER .DEF FILE FOR Am2903-2904-2910

- AM2903.DEF listing
- AM2903.SRC listing
- SAMPLE CODE FOR THE Am2903-2910 from the 2900 Family Study Guide

AM2903.DEF

```
TITLE EXAMPLE DEFINITION FILE FOR THE AM2903/29203
;
;
WORD    64
;
; AM2903 INSTRUCTION SET
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; ALU SOURCE OPERANDS (EA, I0, OEB)
; 16 REGISTER - TWO ADDRESS VERSION
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
RAMAB:   EQU     Q#0             ; RAM A PORT, RAM B PORT
RAMADB:  EQU     Q#1             ; RAM A PORT, DATA BUS B
RAMAQ:   EQU     Q#2             ; OR Q#3 - RAM A PORT, Q REGISTER
DARAMB:  EQU     Q#4             ; DATA BUS A, RAM B PORT
DADB:    EQU     Q#5             ; DATA BUS A, DATA BUS B
DAQ:     EQU     Q#6             ; OR Q#7 - DATA BUS A, Q REGISTER
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; ALU FUNCTIONS - NORMAL MODE (I4, I3, I2, I1, I0 ALL NOT 0)
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
SPECL:   EQU     H#0             ; I0 MUST BE LOW
HIGH:    EQU     H#0             ; I0 MUST BE HIGH (Q REGISTER SELECT)
SUBR:    EQU     H#1             ; F = S - R - 1 + Cin
SUBS:    EQU     H#2             ; F = R - S - 1 + Cin
ADD:     EQU     H#3             ; F = R + S + Cin
INCRS:   EQU     H#4             ; F = S + Cin
INCSNON:EQU      H#5             ; F = NOT S + Cin
INCRR:   EQU     H#6             ; F = R + Cin
INCRNON:EQU      H#7             ; F = NOT R + Cin
LOW:     EQU     H#8             ; F = LOW
NOTRS:   EQU     H#9             ; F = NOT R AND S
EXNOR:   EQU     H#A             ; F = R EXNOR S
EXOR:    EQU     H#B             ; F = R EXOR S
AND:     EQU     H#C             ; F = R AND S
NOR:     EQU     H#D             ; F = R NOR S
NAND:    EQU     H#E             ; F = R NAND S
OR:      EQU     H#F             ; F = R OR S
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * *
;
; ALU DESTINATION CONTROL ( I8 - I7 - I6 - I5)
;     NORMAL FUNCTIONS
; * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
RAMDA:   EQU      H#0              ; F TO RAM, ARITHMETIC DOWN SHIFT
RAMDL:   EQU      H#1              ; F TO RAM, LOGICAL DOWN SHIFT
RAMQDA:  EQU      H#2              ; DOUBLE PRECISION ARITHMETIC DOWN SHIFT
RAMQDL:  EQU      H#3              ; DOUBLE PRECISION LOGICAL DOWN SHIFT
RAM:     EQU      H#4              ; F TO RAM WITH PARITY
QD:      EQU      H#5              ; F TO Y, DOWN SHIFT Q
LOADQ:   EQU      H#6              ; F TO Q WITH PARITY
RAMQ:    EQU      H#7              ; F TO RAM AND Q WITH PARITY
RAMUPA:  EQU      H#8              ; F TO RAM, ARITHMETIC UP SHIFT
RAMUPL:  EQU      H#9              ; F TO RAM, LOGICAL UP SHIFT
RAMQUPA:EQU       H#A              ; DOUBLE PRECISION ARITHMETIC UP SHIFT
RAMQUPL:EQU       H#B              ; DOUBLE PRECISION LOGICAL UP SHIFT
YBUS:    EQU      H#C              ; F TO Y ONLY
QUP:     EQU      H#D              ; F TO Y, UP SHIFT Q
SIGNEXT:EQU       H#E              ; SIO0 TO Yi
RAMEXT:  EQU      H#F              ; F TO Y, SIGN EXTEND LEAST SIG. BYTE
;
; * * * * * * * * * * * * * * ** * * * * * * * * * * * *
;
; SPECIAL FUNCTIONS (I8-I7-I6-I5)
;
;
; Am2903 FUNCTIONS ONLY
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
MULT:    EQU      H#0              ; UNSIGNED MULTIPLY
TWOMULT:EQU       H#2              ; TWO'S COMPLEMENT MULTIPLY
TWOLAST:EQU       H#6              ; TWO'S COMPLEMENT MULTIPLY LAST STEP
INCRMNT:EQU       H#4              ; INCREMENT BY 1 + Cin
SGNTWO:  EQU      H#5              ; SIGN MAGNITUDE-TWO'S COMPL CONVERTION
SLN:     EQU      H#8              ; SINGLE LENGTH NORMALIZE
DLN:     EQU      H#A              ; DOUBLE LENGTH NORMALIZE
DIVFRST:EQU       H#A              ; TWO'S COMPLEMENT DIVIDE FIRST STEP
DIVIDE:  EQU      H#C              ; TWO'S COMPLEMENT DIVIDE MIDDLE STEPS
DIVLAST:EQU       H#E              ; TWO'S COMPLEMENT DIVIDE LAST STEP
;
;
;
; Am29203 ADDITIONS FORTHCOMING
;
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
; DEFINITION FILE FOR FIGURE 29, PAGE 2-57, 1980 DATA BOOK
;
; EXPANDED MEMORY FOR THE Am2903 USING THE Am20705
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; ONLY THE SOURCE FIELDS CHANGE - EXPANDED
; A THIRD ADDRESS FIELD WAS ALSO ADDED (ADDRESS C)
;
;
;
; SOURCE OPERANDS
; ADDED A ADDRESS BITS (A6-A5-A4)
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
AINALU: EQU     Q#0             ; A ADDRESSES 2903 REGISTERS
AIS7051:EQU     Q#1             ; A ADDRESSES FIRST 29705 ADDITION
AIS7052:EQU     Q#2             ; A ADDRESSES SECOND 29705 ADDITION
ACONST: EQU     Q#3             ; A ADDRESSES CONSTANT PROM
ABUS:   EQU     Q#4             ; A FROM BUS
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; ADDED B ADDRESS BITS (B5-B4)
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
BINALU: EQU     B#00            ; B ADDRESSES 2903 REGISTERS
BIS7051:EQU     B#01            ; B ADDRESSES FIRST 29705 ADDTION
BIS7052:EQU     B#10            ; B ADDRESSES SECOND 29705 ADDITION
BBUS:   EQU     B#11            ; B FROM BUS
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; THREE ADDRESS OPERATION - THIRD ADDRESS FIELD
; ADDED C ADDRESS BITS (C5-C4)
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
CIN2903:EQU     B#00            ; C ADDRESSES 2903 REGISTERS
CIS7051:EQU     B#01            ; C ADDRESSES FIRST 29705 ADDITION
CIS7052:EQU     B#10            ; C ADDRESSES SECOND 29705 ADDITION
CBUS:   EQU     B#11            ; C TO B BUS OUT
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; IO SOURCE SELECT FIELD REPLACES EA-IO-OEB THREE-BIT FIELD
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
QREGSEL:EQU     B#1             ; SOURCE IS Q REGISTER
NONQREG:EQU     B#0             ; SOURCE IS RAMB OR B.BUS
;
;
; * * * * * * * * * * * * * * * * * * * *
;
; MISCELLANEOUS FIELDS
;
; REGISTERS
; * * * * * * * * * * * * * * * * * * * *
;
R0:             EQU     H#0
R1:             EQU     H#1
R2:             EQU     H#2
R3:             EQU     H#3
R4:             EQU     H#4
R5:             EQU     H#5
R6:             EQU     H#6
R7:             EQU     H#7
R8:             EQU     H#8
R9:             EQU     H#9
R10:            EQU     H#A
R11:            EQU     H#B
R12:            EQU     H#C
R13:            EQU     H#D
R14:            EQU     H#E
R15:            EQU     H#F
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; CARRY BIT (2 BITS FOR NOW)
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
CARRY:          EQU     B#01
NOCARRY:        EQU     B#00            ;IMAGINATIVE!
IC:             EQU     B#10            ; Cin is Cout
Z:              EQU     B#11            ; Z is Cin
;
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; Am2910 MICROPROGRAM CONTROLLER INSTRUCTION SET
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
JZ:             EQU     H#0             ; RESET STACK, MICROPC, ADDRESS
CJS:            EQU     H#1             ; COND JUMP SUBROUTINE, PUSH STACK
JMAP:           EQU     H#2             ; UNCOND JUMP TO MEMORY MAP (Di)
CJP:            EQU     H#3             ; COND JUMP PIPELINE
PUSH:           EQU     H#4             ; PUSH STACK, LOAD REG MAYBE, CONT
JSRP:           EQU     H#5             ; JUMP SUB FROM REG (F) OR PIPE(T)
CJV:            EQU     H#6             ; COND JUMP TO VECTOR INTER (Di)
JRP:            EQU     H#7             ; JUMP TO REG (F) OR PIPE (T)
RFCT:           EQU     H#8             ; DO LOOP REPEAT UNTIL CTR=0 - STACK
RPCT:           EQU     H#9             ; DO LOOP UNTIL CTR=0 - PIPE
CRTN:           EQU     H#A             ; COND RETURN, POP STACK (T)
CJPP:           EQU     H#B             ; COND JUMP PIPELINE, POP STACK
LDCT:           EQU     H#C             ; LOAD REGISTER, CONTINUE
LOOP:           EQU     H#D             ; DO LOOP UNTIL TEST=T - STACK
CONT:           EQU     H#E             ; CONTINUE
TWB:            EQU     H#F             ; THREE WAY (DEAD MAN TIMER!)
;
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; Am2904 SHIFT INSTRUCTIONS (I9-I8-I7-I6 AND SE)
;     I10 IS TIED TO I8 OF Am2903/29203
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
; DOWN SHIFTING
;
SDZRZQ:          EQU     H#0             ; Z->RN; Z->QN
SDOROQ:          EQU     H#1             ; 1->RN; 1->QN
SLN.RECOVER:     EQU     H#2             ; 0->RN; R0->Mc; MN->QN
DDOR:            EQU     H#3             ; 1->RN; R0->QN
DDMCR:           EQU     H#4             ; Mc->RN; R0->QN
DLN.RECOVER:     EQU     H#5             ; MN->RN; R0->QN
DDZR:            EQU     H#6             ; 0->RN; R0->QN
DDZRQMC:         EQU     H#7             ; 0->RN; R0->QN; Q0->Mc
SDROTMC:         EQU     H#8             ; ROT.R; R0->Mc; ROT.Q
SDROTC:          EQU     H#9             ; ROT.R WITH Mc; ROT.Q
SDROT:           EQU     H#A             ; ROT.R; ROT.Q
SDIC:            EQU     H#B             ; Ic->RN; R0->QN
DDROTC:          EQU     H#C             ; Mc->RN; R0->QN; Q0->Mc
DDROTMC:         EQU     H#D             ; Q0->RN; R0->QN; Q0->Mc
DDINIOVR:        EQU     H#E             ; IN EXOR IOVR -> RN; R0->QN
DDROT:           EQU     H#F             ; DOUBLE PRECISION ROTATE DOWN
;
;
; UP SHIFTING
;
SURZQZ:          EQU     H#2             ; R0<-0; Q0<-0
;
;
; SHIFT ENABLES
;
SE.EN:           EQU     B#0             ; ENABLE SHIFTING
SE.DIS:          EQU     B#1             ; DISABLE SHIFTING
;
;
;
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; Am2904 STATUS REGISTER INSTRUCTION CODES
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; MACHINE STATUS REGISTER INSTRUCTION CODES
;    I5-I4-I3-I2-I1-I0 AND EZ-EC-EN-EOVR-CEM ENABLES
; MICRO STATUS REGISTER INSTRUCTION CODES
;    I5-I4-I3-I2-I1-I0 AND CEu ENABLE
;
; THE FOLLOWING TAKES THESE ALL TOGETHER - YOU MAY WISH TO DO THIS ANOTHER WA
;
; ORDER: 543 210 ZCNOVR CEM CEu
;        Q#  Q#  H#     B#  B#
;
ONELEVEL:       EQU     12Q#0000        ; Y -> MSR; MSR -> USR
SET.MSR:        EQU     12Q#0101        ; SET MACRO STATUS ONLY
SET.USR:        EQU     12Q#0176        ; SET MICRO STATUS ONLY
SWAP.REG:       EQU     12Q#0200        ; MSR <--> USR
;
LOAD.MSR:       EQU     12Q#2001        ; ALU STATUS -> MSR
; THE ABOVE IS ON OF SEVERAL CODES - YOU DON'T NEED THEM ALL!
;
LOAD.USR:       EQU     12Q#2076        ; ALU STATUS -> USR
; DITTO!
;
LOAD.BOTH:      EQU     12Q#2000        ; ALU -> MSR, USR
; AGAIN DITTO!
;
LDINVRTM:       EQU     12Q#3001        ; ALU -> MSR; Ic INVERTED
LDINVRTU:       EQU     12Q#3076        ; ALU -> USR; Ic INVERTED
LOAD.INVERT:    EQU     12Q#3000        ; ALU -> MSR, USR; Ic INVERTED
;
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; Am2904 CONDITION CODE OUTPUT INSTRUCTION CODES
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; caution!  I5-I4-I3-I2-I1-I0 ARE ALSO USED FOR TESTING!!!!
; ENABLE TESTING VIA OEct ENABLE
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
TESTMZ:          EQU      12Q#4477        ; NO STATUS OPERATION
TESTMOVR:        EQU      12Q#4677        ; NO STATUS OPERATION
TESTMC:          EQU      12Q#5277        ; NO STATUS OPERATION
TESTMN:          EQU      12Q#5677        ;
TEST.IOVR:       EQU      12Q#6677        ;
TEST.IC:         EQU      12Q#7277        ;
;
;
; TEST ENABLE
;
OECTEN:          EQU      B#0
OECTDIS:         EQU      B#1
;
;
; OUTPUT ENABLE
;
OEYEN:           EQU      B#0
OEYDIS:          EQU      B#1
;
; INSTRUCTION ENABLE
;
IEN:             EQU      B#0
IENDIS:          EQU      B#1
;
;
; CONDITIONAL CODE MULTIPLEXER (DATA MONITOR)
;
NOACK:           EQU      Q#0
COUT:            EQU      Q#1
PASS:            EQU      Q#7
;
;
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; TWO ADDRESS OPERATION - NO EXPANDED MEMORY
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
AM2903:  DEF 19X, 3VQ#0, 4VH#F, 4VH#C, 2VB#00, 4VH#0, 4VH#0, 1VB#0, 1VB#0, 22
;  DEFAULTS       RAMAB  OR     YBUS   NOCin    R0      R0      IEN     OEY.EN
;
;
AM2910:   DEF   4VH#E, 3VX, 12V$X, 45X
; DEFAULTS      CONT   #     #
;
AM2904:   DEF   42X,     12VQ#2001, 1VB#1,  1VB#0, 4VX,  1VB#1, 3X
; DEFAULTS                LOAD.MSR   OECTDIS OEYEN  X     SE.DIS
;
SHIFT:    DEF   56X,  4VX, 1B#0, 3X
;                     SHIFT SE.EN
TEST:     DEF   42X, 12VQ#7777, 1VB#0, 9X
;                    DISABLED   OECTEN
STATUS:   DEF   42X, 12VQ#2001, B#1, 1VB#0, 4X, B#1, 3X
;                    LOAD.MSR  NO CT OEYEN     SE.DIS

;
; ADDED STATEMENTS FOR DATA MONITOR PROBLEM
;
NOP2903:  DEF   19X, Q#0, H#F, H#C, B#00, H#0, H#0, B#0, B#1, 22X
;
CTRL:     DEF   61X, 1VB#0, 1VB#0, 1VB#0
;                    DATAin DATAout MEMORY MAP SELECT
;                    CTRL   CTRL    FIRST QUADRANT
;
;
;
END
```

```
TITLE TEST FILE FOR AM2903 .DEF FILE

; ------------------------------------------------------------------
;                PARTIAL MICROWORD ONLY !!!!!!!!
;                       DEMO FILE !
; ------------------------------------------------------------------
;
;        SAMPLE CODE  Am2903-Am2910
;
; * * * * *
; SAMPLE Am2903 OPERATIONS FROM THE ED2900A CLASS NOTES
;                               THE 2900 FAMILY STUDY GUIDE
;                               THE ED2900B CLASS NOTES
; * * * * *
;-------------------------------------------------------------------
; 15. DA + DB --> Yi
;-------------------------------------------------------------------
        AM2910 & AM2903 DADB, ADD, YBUS, NOCARRY
;-------------------------------------------------------------------
; 16. RA + RB --> RC (ANY THREE REGISTERS)
;-------------------------------------------------------------------
        AM2910 & AM2903 , ADD, RAM, NOCARRY, R0, R1      ; ADD THIRD REG FIELD
;-------------------------------------------------------------------
; 18.  INCREMENT R15 AND OUTPUT ITS ORIGINAL VALUE
;-------------------------------------------------------------------
        AM2910 & AM2903 , INCRR, RAM, CARRY, R15, R15
;-------------------------------------------------------------------
; 17.  FIRMWARE BYTE SWAP
;-------------------------------------------------------------------
        AM2910 LDCT,, H#2 & AM2903 , ADD, RAMUPL, IC, R15, R15 & SHIFT SDROT
LA:     AM2910 RPCT,, LA  & AM2903 , ADD, RAMUPL, IC, R15, R15 & SHIFT SDROT
;-------------------------------------------------------------------
; HARDWARE-ASSISTED BYTE SWAP
;-------------------------------------------------------------------
        AM2910 & AM2903 DARAMB, INCRR, RAM, NOCARRY, , R15
; OR...
        AM2910 & AM2903 RAMAB,        , RAM,         , , R15, , OEYDIS
;-------------------------------------------------------------------
; 19. DA --> Q
;-------------------------------------------------------------------
        AM2910 & AM2903 DARAMB, INCRR, LOADQ, NOCARRY
;-------------------------------------------------------------------
; 20. OUTPUT R2 AND PERFORM 2*(R2+1) --> R2 IN ONE MICROCYCLE
;-------------------------------------------------------------------
        AM2910 & AM2903 , INCRR, RAMUPL, CARRY, R2, R2 & SHIFT SURZQZ
; OR...
        AM2910 & AM2903 , INCRR, RAMUPA, CARRY, R2, R2 & SHIFT SURZQZ
;-------------------------------------------------------------------
; 21. UNSIGNED 16 BIT MULTIPLY (R1*R2)
;-------------------------------------------------------------------
        AM2910 LDCT ,, H#F & AM2903 , INCRR, LOADQ, NOCARRY, R2
LB:     AM2910 RPCT ,, LB  & AM2903 , SPECL, MULT,  NOCARRY, R1, R0
```

```
;-------------------------------------------------------------------------
; 22. TWO'S COMPLEMENT 16 BIT MULTIPLY (R1*R2)
;-------------------------------------------------------------------------
        AM2910 LDCT ,, H#E & AM2903 , INCRR, LOADQ,   NOCARRY, R2
LC:     AM2910 RPCT ,, LC  & AM2903 , SPECL, TWOMULT, NOCARRY, R1, R0
/&              SHIFT DDOR
        AM2910             & AM2903 , SPECL, TWOLAST, Z,       R1, R0
/&              SHIFT DDOR
;-------------------------------------------------------------------------
; 23. PERFORM A DOUBLE PRECISION DOWN SHIFT USING R2 AND Q
;-------------------------------------------------------------------------
        AM2910 & AM2903 , INCRS, RAMQDL, , , R2 & SHIFT DDZR
;-------------------------------------------------------------------------
; 24. PERFORM 4*R2 --> Q IN ONE MICROCYCLE
;-------------------------------------------------------------------------
        AM2910 & AM2903 , ADD, RAMQUPA, NOCARRY, R2, R2 & SHIFT SURZQZ
        AM2910 & AM2903 , INCRS, LOADQ, NOCARRY,    , R2
; ********** REQUIRES TWO MICROCYCLES! ********************************
;-------------------------------------------------------------------------
; 27. SINGLE LENGTH NORMALIZE OF R6 (16 BIT ALU)
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
; 28. DOUBLE LENGTH NORMALIZE OF R6.R7 (16 BIT ALU)
;-------------------------------------------------------------------------


;
END
```

LAB FOUR

DATA MONITOR DEF & SRC FILES

LAB FOUR - THE DATA MONITOR .DEF AND .SRC FILES

YOUR HOMEWORK LAST NIGHT WAS TO CREATE THE .DEF AND .SRC FILES FOR THE DATA MONITOR 2910-2901 DESIGN PROBLEM (ED2900A)

YOU HAVE UNTIL 10:30 AM TO COMPLETE YOUR FILE CREATION

LABS FIVE AND SIX WILL ASSUME THAT YOU HAVE DONE SO!

DATA MONITOR

THE DATA MONITOR

- MONITOR2.SRC listing
  (uses AM2903.DEF)
- MONITOR.P2L AMDASM ASSEMBLY listing
  - .SRC SEQUENCED
  - ENTRY POINTS
  - CONTROL MEMORY PROM CONTENTS
    (X for Don't Cares)
  - SYMBOLS listing
- MONITOR.OPC listing
- AMMAP ASSEMBLY listing
- .OPC SEQUENCED
- ENTRY POINTS relisted
- MEMORY MAP CONTENTS

```
TITLE DATA MONITOR - Am2903 VERSION
;
;
;
; ADDED EQUATES FOR MONITOR
; ( USING AM2903.DEF FILE )
;
LOAD:           EQU     B#1
NOLOAD:         EQU     B#0
;
FIVES:          EQU     B#1
ZRANGE:         EQU     B#0
;
; * * * * * * * * * * * * * * * *
; CODE TO INITIALIZE REGISTERS
; R0=R1=R2=R3=R4 <-- 0
; * * * * * * * * * * * * * * * *
;
START:  AM2910 & AM2903 , EXOR, RAM, , R0, R0
        AM2910 & AM2903 , EXOR, RAM, , R1, R1
        AM2910 & AM2903 , EXOR, RAM, , R2, R2
        AM2910 & AM2903 , EXOR, RAM, , R3, R3
        AM2910 & AM2903 , EXOR, RAM, , R4, R4
;
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; THE ABOVE LINE AND THE BELOW LINE CANNOT BE COMBINED DUE TO THE
; BRANCH ENTRY POINT ( BRANCHING TO NEXT DOES NOT ALWAYS INCLUDE
; RESETTING REGISTER 4)
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;
NEXT:   AM2910 & NOP2903 & CTRL LOAD                    ; LOAD DATAin
;
        AM2910 JMAP & NOP2903                           ; CASE BRANCH
;--------------------------------
; IF DATAin = 0 THEN INCR R0, etc.
;--------------------------------
DISZ::  AM2910 & AM2903 DARAMB, INCRR, RAM, CARRY, R0, R0
        AM2910 CJS, COUT,  SUB0  & NOP2903
        AM2910 & AM2903 ,     , RAM, , R1, R1 & CTRL , LOAD
WAITA:  AM2910 CJP, NOACK, WAITA & NOP2903
        AM2910 CJP, PASS,  CASE  & NOP2903    & CTRL ,     , FIVES
;--------------------------------
; IF DATAin > 0 THEN INCR R1, etc.
;--------------------------------
DGRZ::  AM2910 & AM2903 DARAMB, INCRR, RAM, CARRY, R1, R1
        AM2910 CJS, COUT,  SUB0  & NOP2903
        AM2910 & AM2903 ,     , RAM, , R0, R0 & CTRL , LOAD
WAITB:  AM2910 CJP, NOACK, WAITB & NOP2903    & CTRL ,     , FIVES
;
;-----------------------------------------------------------
; this case statement uses the upper half of the memory map
;-----------------------------------------------------------
;
CASE:   AM2910 JMAP & NOP2903                   & CTRL ,     , FIVES
```

.SRC File

```
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * *
; INCREMENT R2 OR R3 OR R4 BASED ON RANGE OF DATAin
; * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
L5::    AM2910 CJP, PASS,  NEXT  & AM2903 DARAMB, INCRR, RAM, CARRY, R2, R2
GR5L10::AM2910 CJP, PASS,  NEXT  & AM2903 DARAMB, INCRR, RAM, CARRY, R3, R3
GR10::  AM2910 CJP, PASS,  NEXT  & AM2903 DARAMB, INCRR, RAM, CARRY, R4, R4
;
; ------------------------------------------------------------
; SUBROUTINE SUB0
; ------------------------------------------------------------
;
; output R2, R3, R4 and then reset them all
; wait for an ACK after each send
;
;
;
SUB0:   AM2910 & AM2903 ,     , RAM, , R2, R2 & CTRL , LOAD
WAIT1:  AM2910 CJP, NOACK, WAIT1 & NOP2903
        AM2910 & AM2903 ,     , RAM, , R3, R3 & CTRL , LOAD
WAIT2:  AM2910 CJP, NOACK, WAIT2 & NOP2903
        AM2910 & AM2903 ,     , RAM, , R4, R4 & CTRL , LOAD
WAIT3:  AM2910 CJP, NOACK, WAIT3 & NOP2903
        AM2910             & AM2903 , EXOR, RAM, , R2, R2
        AM2910             & AM2903 , EXOR, RAM, , R3, R3
        AM2910 CRTN, PASS & AM2903 , EXOR, RAM, , R4, R4
;
;
;
;
;
END
```

PHASE 2 ASSEMBLY

```
      TITLE DATA MONITOR - Am2903 VERSION
      ;
      ;
      ;
      ; ADDED EQUATES FOR MONITOR
      ; ( USING AM2903.DEF FILE )
      ;
0001  LOAD:           EQU     B#1
      NOLOAD:         EQU     B#0
0000  ;
0001  FIVES:          EQU     B#1
      ZRANGE:         EQU     B#0
      ;
      ;
      ; CODE TO INITIALIZE REGISTERS
      ;
0000  ;
0000  START:  AM2910 & AM2903 , EXOR, RAM, , R0, R0
0001          AM2910 & AM2903 , EXOR, RAM, , R1, R1
0002          AM2910 & AM2903 , EXOR, RAM, , R2, R2
0003          AM2910 & AM2903 , EXOR, RAM, , R3, R3
0004          AM2910 & AM2903 , EXOR, RAM, , R4, R4
0005  NEXT:   AM2910 & NOP2903 & CTRL LOAD
0006          AM2910 JMAP & NOP2903
0007  DISZ::  AM2910 & AM2903 DARAMB, INCRR, RAM, CARRY, R0, R0
0008          AM2910 CJS, COUT,  SUB0  & NOP2903
0009          AM2910 & AM2903 ,      , RAM, , R1, R1 & CTRL , LOAD
000A  WAITA:  AM2910 CJP, NOACK, WAITA & NOP2903
000B          AM2910 CJP, PASS,  CASE  & NOP2903    & CTRL ,      , FIVES
000C  DGRZ::  AM2910 & AM2903 DARAMB, INCRR, RAM, CARRY, R1, R1
000D          AM2910 CJS, COUT,  SUB0  & NOP2903
000E          AM2910 & AM2903 ,      , RAM, , R0, R0 & CTRL , LOAD
000F  WAITB:  AM2910 CJP, NOACK, WAITB & NOP2903    & CTRL ,      , FIVES
0010  CASE:   AM2910 JMAP & NOP2903                 & CTRL ,      , FIVES
0011  L5::    AM2910 CJP, PASS,  NEXT  & AM2903 DARAMB, INCRR, RAM, CARRY, R2, R2
0012  GR5L10::AM2910 CJP, PASS,  NEXT  & AM2903 DARAMB, INCRR, RAM, CARRY, R3, R3
0013  GR10::  AM2910 CJP, PASS,  NEXT  & AM2903 DARAMB, INCRR, RAM, CARRY, R4, R4
      ;
      ; SUBROUTINE SUB0
      ;
0014  SUB0:   AM2910 & AM2903 ,      , RAM, , R2, R2 & CTRL , LOAD
0015  WAIT1:  AM2910 CJP, NOACK, WAIT1 & NOP2903
0016          AM2910 & AM2903 ,      , RAM, , R3, R3 & CTRL , LOAD
0017  WAIT2:  AM2910 CJP, NOACK, WAIT2 & NOP2903
0018          AM2910 & AM2903 ,      , RAM, , R4, R4 & CTRL , LOAD
0019  WAIT3:  AM2910 CJP, NOACK, WAIT3 & NOP2903
001A          AM2910 & AM2903 , EXOR, RAM, , R2, R2
001B          AM2910 & AM2903 , EXOR, RAM, , R3, R3
001C          AM2910 & AM2903 , EXOR, RAM, , R4, R4
001D          AM2910 CRTN, PASS        & NOP2903
      ;
      ;
      ;
      ;
      ;
      END
```

```
DATA MONITOR - AM2903 VERSION

 ENTRY POINTS

  DGRZ      000C
  DISZ      0007
  GR10      0013
  GR5L10    0012
  L5        0011
```

AMDOS/29 AMDASM MICRO ASSEMBLER, V1.0
DATA MONITOR - AM2903 VERSION

```
0000 1110XXXXXXXXXXX XXX0001011010000 0000000000XXXXX XXXXXXXXXXXXXXX
0001 1110XXXXXXXXXXX XXX0001011010000 0001000100XXXXX XXXXXXXXXXXXXXX
0002 1110XXXXXXXXXXX XXX0001011010000 0010001000XXXXX XXXXXXXXXXXXXXX
0003 1110XXXXXXXXXXX XXX0001011010000 0011001100XXXXX XXXXXXXXXXXXXXX
0004 1110XXXXXXXXXXX XXX0001011010000 0100010000XXXXX XXXXXXXXXXXXXXX
0005 1110XXXXXXXXXXX XXX0001111110000 0000000001XXXXX XXXXXXXXXXXX100
0006 0010XXXXXXXXXXX XXX0001111110000 0000000001XXXXX XXXXXXXXXXXXXXX
0007 1110XXXXXXXXXXX XXX1000110010001 0000000000XXXXX XXXXXXXXXXXXXXX
0008 0001001000000010 1000001111110000 0000000001XXXXX XXXXXXXXXXXXXXX
0009 1110XXXXXXXXXXX XXX0001111010000 0001000100XXXXX XXXXXXXXXXXX010
000A 0011000000000001 0100001111110000 0000000001XXXXX XXXXXXXXXXXXXXX
000B 0011111000000010 0000001111110000 0000000001XXXXX XXXXXXXXXXXX001
000C 1110XXXXXXXXXXX XXX1000110010001 0001000100XXXXX XXXXXXXXXXXXXXX
000D 0001001000000010 1000001111110000 0000000001XXXXX XXXXXXXXXXXXXXX
000E 1110XXXXXXXXXXX XXX0001111010000 0000000000XXXXX XXXXXXXXXXXX010
000F 0011000000000001 1110001111110000 0000000001XXXXX XXXXXXXXXXXX001
0010 0010XXXXXXXXXXX XXX0001111110000 0000000001XXXXX XXXXXXXXXXXX001
0011 0011111000000000 1011000110010001 0010001000XXXXX XXXXXXXXXXXXXXX
0012 0011111000000000 1011000110010001 0011001100XXXXX XXXXXXXXXXXXXXX
0013 0011111000000000 1011000110010001 0100010000XXXXX XXXXXXXXXXXXXXX
0014 1110XXXXXXXXXXX XXX0001111010000 0010001000XXXXX XXXXXXXXXXXX010
0015 0011000000000010 1010001111110000 0000000001XXXXX XXXXXXXXXXXXXXX
0016 1110XXXXXXXXXXX XXX0001111010000 0011001100XXXXX XXXXXXXXXXXX010
0017 0011000000000010 1110001111110000 0000000001XXXXX XXXXXXXXXXXXXXX
0018 1110XXXXXXXXXXX XXX0001111010000 0100010000XXXXX XXXXXXXXXXXX010
0019 0011000000000011 0010001111110000 0000000001XXXXX XXXXXXXXXXXXXXX
001A 1110XXXXXXXXXXX XXX0001011010000 0010001000XXXXX XXXXXXXXXXXXXXX
001B 1110XXXXXXXXXXX XXX0001011010000 0011001100XXXXX XXXXXXXXXXXXXXX
001C 1110XXXXXXXXXXX XXX0001011010000 0100010000XXXXX XXXXXXXXXXXXXXX
001D 1010111XXXXXXXXX XXX0001111110000 0000000001XXXXX XXXXXXXXXXXXXXX
```

DATA MONITOR - AM2903 VERSION

SYMBOLS

ABUS	0004
ACONST	0003
ADD	0003
AINALU	0000
AIS7051	0001
AIS7052	0002
AND	000C
BBUS	0003
BINALU	0000
BIS7051	0001
BIS7052	0002
CARRY	0001
CASE	0010
CBUS	0003
CIN2903	0000
CIS7051	0001
CIS7052	0002
CJP	0003
CJPP	000B
CJS	0001
CJV	0006
CONT	000E
COUT	0001
CRTN	000A
DADB	0005
DAQ	0006
DARAMB	0004
DDINIOVR	000E
DDMCR	0004
DDOR	0003
DDROT	000F
DDROTC	000C
DDROTMC	000D
DDZR	0006
DDZRQMC	0007
DGRZ	000C
DISZ	0007
DIVFRST	000A
DIVIDE	000C
DIVLAST	000E
DLN	000A
DLN.RECO	0005
EXNOR	000A
EXOR	000B
FIVES	0001
GR10	0013
GR5L10	0012
HIGH	0000
IC	0002
IEN	0000
IENDIS	0001
INCRMNT	0004
INCRNON	0007
INCRR	0006
INCRS	0004
INCSNON	0005
JMAP	0002
JRP	0007
JSRP	0005
JZ	0000
L5	0011
LDCT	000C
LDINVRTM	0601
LDINVRTU	063E
LOAD	0001
LOAD.BOT	0400
LOAD.INV	0600
LOAD.MSR	0401
LOAD.USR	043E
LOADQ	0006
LOOP	000D
LOW	0008
MULT	0000
NAND	000E
NEXT	0005
NOACK	0000
NOCARRY	0000
NOLOAD	0000
NONQREG	0000
NOR	000D
NOTRS	0009
OECTDIS	0001
OECTEN	0000
OEYDIS	0001
OEYEN	0000
ONELEVEL	0000
OR	000F
PASS	0007
PUSH	0004
QD	0005
QREGSEL	0001
QUP	000D
R0	0000
R1	0001
R10	000A
R11	000B
R12	000C
R13	000D
R14	000E
R15	000F
R2	0002
R3	0003
R4	0004
R5	0005
R6	0006
R7	0007
R8	0008
R9	0009
RAM	0004
RAMAB	0000
RAMADB	0001
RAMAQ	0002
RAMDA	0000
RAMDL	0001
RAMEXT	000F
RAMQ	0007
RAMQDA	0002
RAMQDL	0003
RAMQUPA	000A
RAMQUPL	000B
RAMUPA	0008
RAMUPL	0009
RFCT	0008
RPCT	0009
SDIC	000B
SDOROQ	0001
SDROT	000A
SDROTC	0009
SDROTMC	0008
SDZRZQ	0000
SE.DIS	0001
SE.EN	0000
SET.MSR	0041
SET.USR	007E
SGNTWO	0005
SIGNEXT	000E
SLN	0008
SLN.RECO	0002
SPECL	0000
START	0000
SUB0	0014
SUBR	0001
SUBS	0002
SURZQZ	0002
SWAP.REG	0080
TEST.IC	0EBF
TEST.IOV	0DBF
TESTMC	0ABF
TESTMN	0BBF
TESTMOVR	09BF
TESTMZ	093F
TWB	000F
TWOLAST	0006
TWOMULT	0002
WAIT1	0015
WAIT2	0017
WAIT3	0019
WAITA	000A
WAITB	000F
YBUS	000C
Z	0003
ZRANGE	0000

TOTAL PHASE 2 ERRORS = 0

.OPC FILE

```
TITLE DATA MONITOR MAPPING PROM
;
;
;
WIDTH 8
;
;
        DISZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        DGRZ
        L5
        L5
        L5
        L5
        L5
        L5
        GR5L10
        GR5L10
        GR5L10
        GR5L10
        GR5L10
        GR10
        GR10
        GR10
        GR10
        GR10
;
END
```

AMMAP ASSEMBLY

```
AMDOS/29 AMMAP ASSEMBLER, V1.1
DATA MONITOR MAPPING PROM

      ;
      ;
      ;
      WIDTH 8
      ;
      ;
0000          DISZ
0001          DGRZ
0002          DGRZ
0003          DGRZ
0004          DGRZ
0005          DGRZ
0006          DGRZ
0007          DGRZ
0008          DGRZ
0009          DGRZ
000A          DGRZ
000B          DGRZ
000C          DGRZ
000D          DGRZ
000E          DGRZ
000F          DGRZ
0010          L5
0011          L5
0012          L5
0013          L5
0014          L5
0015          L5
0016          GR5L10
0017          GR5L10
0018          GR5L10
0019          GR5L10
001A          GR5L10
001B          GR10
001C          GR10
001D          GR10
001E          GR10
001F          GR10
      ;
      END
```

```
AMDOS/29 AMMAP ASSEMBLER, V1.1
DATA MONITOR MAPPING PROM

 ENTRY POINT SYMBOLS

  DGRZ      000C
  DISZ      0007
  GR10      0013
  GR5L10    0012
  L5        0011

 TOTAL ASSEMBLY ERRORS =    0
```

```
AMDOS/29 AMMAP ASSEMBLER, V1.1
DATA MONITOR MAPPING PROM

 0000 00000111 (0007)
 0001 00001100 (000C)
 0002 00001100 (000C)
 0003 00001100 (000C)
 0004 00001100 (000C)
 0005 00001100 (000C)
 0006 00001100 (000C)
 0007 00001100 (000C)
 0008 00001100 (000C)
 0009 00001100 (000C)
 000A 00001100 (000C)
 000B 00001100 (000C)
 000C 00001100 (000C)
 000D 00001100 (000C)
 000E 00001100 (000C)
 000F 00001100 (000C)
 0010 00010001 (0011)
 0011 00010001 (0011)
 0012 00010001 (0011)
 0013 00010001 (0011)
 0014 00010001 (0011)
 0015 00010001 (0011)
 0016 00010010 (0012)
 0017 00010010 (0012)
 0018 00010010 (0012)
 0019 00010010 (0012)
 001A 00010010 (0012)
 001B 00010011 (0013)
 001C 00010011 (0013)
 001D 00010011 (0013)
 001E 00010011 (0013)
 001F 00010011 (0013)
```

LAB FIVE:

LBPM

SBPM

RBPM

VBPM

& DDT29

LAB FIVE - WCS

- POWER UP THE SYSTEM

- BE CERTAIN THAT THE CLOCK IS OFF

- SET DIP SWITCH ON WRITABLE CONTROL STORE CARD SO THAT THE FIRST THREE FROM THE TOP ARE PUSHED TOWARDS THE REAR OF THE CARD OR CHASSIS

    - DO NOT SHORT OUT THE CARDS
    - USE A FINGER OR OTHER NON-ABRASIVE DEVICE TO MOVE THEM

START WITH THE DATA MONITOR SRC FILE OUTPUT

- LOAD WCS (LOWER HALF OF MICROWORD)

  LBPM B:MONITOR WCS CL

- SIGN ON TO DDT29 VIA

  A>DDT29

- SYSTEM PROMPT IS NOW A "."

- TYPE

  D

- YOU SHOULD BE LOOKING AT THE FIRST 16 MICROWORDS IN THE WCS

- TYPE

  D 0 1DPc

- YOU SHOULD SEE AND PRINT OUT THE FIRST 6 MICROWORDS

- TURN PRINTER OFF

  Pc

- TYPE

  S 5 C BBBB

  D 5 5

- YOU HAVE JUST SUBSTITUTED BBBB INTO BYTES C AND D OF MICROWORD AT ADDRESS 5

- TYPE

  E

- THIS EXITS DDT29

- TYPE

  SBPM B:FILE1 WC FR 0 TO 1D

- THIS WILL SAVE THE ALTERED FILE AS FILE1.SVW

- TYPE

  RBPM B:FILE1.SVW

- RELOADS THE WCS

- TYPE

  VBPM B:MONITOR WC FR 0 TO 1D CL

- COMPARES PATCHED FILES WITH THE WCS

  HELPS YOU REMEMBER PATCHES, ETC

- TYPE

  LBPM B:MONITOR WCS WA 1E CL DC 1

  DDT29

  D 0 3BPc

Microprogramming Support Software Programs

Program	Format
DDT29 (Dynamic Debugging Tool-2900)	Dw | X.Y / X Y | Display WCS from address X for Y words. Display WCS from address X through address Y. DM | X.Y / X Y | Display mapping memory from address X for Y words. Display mapping memory from address X through address Y. STatus Display Instrumentation Card status register contents. Sw | XYD | Store Hexadecimal data in WCS beginning at address X, byte Y. SM | XD | Store hexadecimal data in mapping memory beginning at address X. Halt Stops Microprogrammed System clock. MS n Micro-step Microprogrammed System clock n (1-65535) steps (1 step = 1 microcycle). (No count = 1.) SS n Single-step Microprogrammed System clock n (1-65535) phases of a multiphase microcycle. Run Runs Microprogrammed System clock. CTL XX Store hexadecimal value 00-FF in Instrumentation Card control register to set mask bits. Jamccu Jam value in Instrumentation Card address register onto microprogram memory address bus. IR X Store hexadecimal value X into Instrumentation Card address register. Z N Sleep N milliseconds. Default value for N is 1. M N ddt29 subcommand, ddt29 subcommand, ...ddt29 subcommand Execute N times the user-specified string of DDT 29 subcommands. Default value for N infinity. Execution of the M subcommand can be terminated at any time by pressing the CRT Console DEL key. DLA Display address of the last microinstruction executed. DMB Display 20 monitor bits of user selected and wired test point or other data at CRT Console. Exit Leave DDT29 and enter AMDOS 29.

Microprogramming Support Software Programs

Program	Format
LBPM (Load Bipolar Memory WCS or CCU Mapping Memory) VBPM (Verify Bipolar Memory Contents Previously Loaded Into WCS or CCU Mapping Memory)	LBPM Filename [WCs / MAp] [FRom X] [TO X / FOr X] [WA X] [LSb d] [NOClear / CLear] [LOwer / UPper / UL] [VErify / NOVerify] [DC b] [SBft] VBPM Filename [WCs / MAp] [FRom X] [TO X / FOr X] [WA X] [LSb d] [NOClear / CLear] [LOwer / UPper / UL] [DC b] Where: Filename is name of diskette file to be loaded into WCS or CCU mapping memory. FRom designates starting PC within diskette file. (Default is starting file PC.) TO designates ending PC within diskette file. (Default is entire file.) FOr designates X number of microwords to load. (Default is entire file.) WA (Writable Address) designates starting address X within WCS or CCU mapping memory. LSb designates least significant bit number d (0-127) of microword. (Default is 0, right justified.) NOClear designates don't clear WCS before loading. (Default is NOClear.) CLear designates clear each WCS word before loading. (Default is NOClear.) UL designates all 128 bits of WCS are accessed. (Default is LOwer.) LOwer designates that only bits 0 to 63 of WCS are accessed. (Default is LOwer.) UPper designates that only bits 64-127 of WCS are accessed. (Default is LOwer.) NOVerify designates that verify phase is not run. (Default is VErify.) VErify designates that verify phase is run automatically. SBft designates given operands are placed into bipolar format table for subsequent default value use. (See BFt portion of SET command.) DCb designates value "don't cares" are set to. (Default is 0.)
SBPM (Save Bipolar Memory) RBPM (Restore Bipolar Memory)	SBPM Filename [WCs / MAp] [FRom X] [TO X / FOr X] RBPM Filename. [SVW / SVM] Where: Filename is name of diskette file created to receive saved WCS microcode or CCU mapping memory contents. Complete filename for saved WCS contents is "filename.SVW". Complete filename for CCU mapping memory contents is "filename.SVM" SBPM prompts with a ". . . . erase (Y or N)" if "filename.SVW" or "filename.SVM" already exists. FRom, TO, and FOr are valid for WCs only. FRom designates starting PC within WCS (Default is 0.) TO designates ending PC within WCS (Default is FRom and results in one microword saved.) FOr designates number of microwords to save (Default is 1.)

LAB 6

AMMAP

AMSCRM

AMPROM

LAB SIX - AMMAP - AMSCRM - AMPROM

AMMAP

- POWER UP SYSTEMS
- CHECK DATA DISK FOR SIMPLE.DEF, SIMPLE.SRC

  A>DIR B:SIMPLE.*
- ASSEMBLE VIA AMDASM (PHASE 1 AND PHASE 2)

  OR LOCATE B:SIMPLE.MAP
- CREATE (OR LOCATE) SIMPLE.OPC

  (CREATE USING THE EDITOR)
- CALL UP AMMAP

  CREATE MAP FOR THE SIMPLE COMPUTER

  AMMAP B:SIMPLE MAP = B:SIMPLE

.OPC file .MAP file

- AMMAP CREATES B:SIMPLE.OBM (PROMS, WCS) AND B:SIMPLE.P4L
- PRINT OUT B:SIMPLE.P4L

AMSCRM

- ASSEMBLE SIMPLE.DEF, SIMPLE.SRC IF YOU HAVE NOT ALREADY DONE SO (AND NOT ERASED RESULTING FILES)

- USE AMSCRM AND MOVE THE BRANCH ADDRESS FIELD TO THE FAR RIGHT OF THE MICROWORD (SWAP IT WITH F BUS CONTROL)

```
A>AMSCRM OLD=B:SIMPLE.OBJ NEW=B:SIMPLE.XOB
```

AMPROM

- CALL UP AMPROM FOR MONITOR OR SIMPLE

  A>AMPROM O B:SIMPLE <-- B:SIMPLE.OBJ ASSUMED

- INPUT THE FOLLOWING
  - DON'T CARES SET TO 0
  - WIDTH OF PROMS USED IS YOUR CHOICE (TRY 8)
  - DEPTH OF PROMS USED IS YOUR CHOICE (TRY 16)
  - PRINT OUT BRANCH ADDRESS PROM(s)
  - PRINT OUT ALL PROMS - FIND THE BRANCH ADDRESS PROMS
- DESIGN YOUR MEMORY SO THAT THE BRANCH ADDRESS FIELD IS IN ITS OWN PROMS
- PRINT OUT B:SIMPLE.P3L

- RENAME B:SIMPLE.XOB TO B:SIMPLE.OBJ (CHANGE THE NAME OF B:SIMPLE.OBJ FIRST OR ERASE IT)
- CALL UP AMPROM AS BEFORE
- USE SAME RESPONSES AS BEFORE TO AMPROM'S QUESTIONS
- PRINT OUT NEW B:SIMPLE.P3L

WHENEVER YOU USE AS EXISTING FILE RATHER THAN CREATE ONE OF YOUR OWN - CAUTION! SOMEONE ELSE MAY HAVE CHANGED NAMES ON THESE FILES

SIMPLE COMPUTER .DEF and .SRC

THE VERY SIMPLE COMPUTER

- SIMPLE.DEF listing
- SIMPLE.SRC listing
- SIMPLE.P2L AMDASM ASSEMBLY listing
  - SRC SEQUENCED
  - CONTROL MEMORY PRINTOUT with ENTRY POINTS
  - SYMBOLS list
- SIMPLE.OPC listing
- AMMAP ASSEMBLY LISTING
  - .OPC SEQUENCED
  - MEMORY MAP CONTENTS
  - ENTRY POINT SYMBOLS (relisted)
- AMPROM ASSEMBLY LISTING
  - PROM MAP (from interactive input)
  - PROM CONTENTS

```
TITLE  THE VERY SIMPLE COMPUTER (ED2900A)
;                                                    .DEF
WORD 24
;
; THIS COMPUTER IS NOT BASED ON ANY PARTICULAR PART!
; IT IS SIMPLY AN EXAMPLE OF .DEF AND .SRC FILE CONSTRUCTION
; AND MEMORY MAP USAGE (JMAP Am2910 INSTRUCTION)
;
;
; SEQUENCE CONTROL (VERY LIMITED!)
;
CONT:           EQU     B#00
JMP:            EQU     B#01            ; UNCONDITIONAL (GOTO)
JIFO:           EQU     B#10            ; CONDITIONAL (IF-THEN GOTO)
JOPC:           EQU     B#11            ; JUMP ON OP CODE (JMAP)
;
; MEMORY CONTROL
;
READ:           EQU     B#00
WRITE:          EQU     B#01
DIS:            EQU     B#10            ; NO MEMORY OPERATION (B#11 ALSO VALID
;
; ALU CONTROL
;
ADD:            EQU     Q#0
SUB:            EQU     Q#1
AND:            EQU     Q#2
OR:             EQU     Q#3
EXOR:           EQU     Q#4
INE:            EQU     Q#5             ; D BUS + 1 TO F
RTF:            EQU     Q#6             ; PASS R BUS
DTF:            EQU     Q#7             ; PASS D BUS
;
; R BUS SOURCE CONTROL
;
DIR:            EQU     B#0             ; DATA-IN TO R BUS
MDOR:           EQU     B#1             ; MEMORY DATA OUT TO R BUS
;
; R BUS DESTINATION CONTROL
;
RIR:            EQU     B#0             ; R BUS TO INSTRUCTION REGISTER
NRIR:           EQU     B#1             ; NOTHING TO INSTR. REG.
;
; D BUS CONTROL
;
ACCD:           EQU     B#0             ; ACCUMULATOR TO D BUS
PCD:            EQU     B#1             ; PC REGISTER TO D BUS
;
; F BUS CONTROL
;
FACC:           EQU     B#00            ; F BUS TO ACC
FPC:            EQU     B#01            ; F BUS TO PC REGISTER
FMAR:           EQU     B#10            ; F BUS TO MAR REGISTER
NFD:            EQU     B#11            ; NO F BUS DESTINATION
;
;
CODE:           DEF     2VX, 12V$X, 2VX, 3VX, 1VX, 1VX, 1VX, 2VX
;       ---> NO DEFAULTS!! <----
;
END
```

```
TITLE SAMPLE .SRC FILE FOR THE SIMPLE COMPUTER (ED2900A)
;
;
;
;
START:  CODE CONT,      ,  DIS,  DTF,      , NRIR,  PCD,  FMAR     ; PC -> MAR
        CODE CONT,      , READ,  RTF, MDOR,  RIR,      ,  FMAR     ; FETCH
        CODE JOPC,      ,  DIS,      ,      , NRIR,      ,  NFD      ; DECODE STEP
LDA::   CODE JMP, FETCH, READ,  RTF, MDOR, NRIR,      ,  FACC     ; LOAD ACC
STO::   CODE JMP, FETCH, WRITE,     ,      , NRIR, ACCD,  NFD      ; STORE ACC
LADD::  CODE JMP, FETCH, READ,  ADD, MDOR, NRIR, ACCD,  FACC ; ACC + MEM
LSUB::  CODE JMP, FETCH, READ,  SUB, MDOR, NRIR, ACCD,  FACC     ; ACC - MEM
LOR::   CODE JMP, FETCH, READ,   OR, MDOR, NRIR, ACCD,  FACC     ; ACC OR MEM
LAND::  CODE JMP, FETCH, READ,  AND, MDOR, NRIR, ACCD,  FACC     ; ACC AND MEM
XOR::   CODE JMP, FETCH, READ, EXOR, MDOR, NRIR, ACCD,  FACC     ; ACC EXOR MEM
INA::   CODE JMP, FETCH,  DIS,  RTF,  DIR, NRIR,      ,  FACC     ; DATA TO ACC
OUT::   CODE JMP, FETCH,  DIS,      ,      , NRIR, ACCD,  NFD      ; ACC TO DATA-OUT
GOTO::  CODE CONT,      ,  DIS,  DTF,      , NRIR,  PCD,  FMAR ; GO TO addr
        CODE JMP, START, READ,  RTF, MDOR, NRIR,      ,  FPC      ; REFETCH INSTR
IF::    CODE JIFO, GOTO,  DIS,      ,      , NRIR,      ,  NFD      ; IF ACC = 0 GO TO addr
FETCH:  CODE JMP, START,  DIS,  ONE,      , NRIR,  PCD,  FPC      ; PC <- PC + 1
;
;
END
```

.SRC

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.0
SAMPLE .SRC FILE FOR THE SIMPLE COMPUTER (ED2900A)

      TITLE SAMPLE .SRC FILE FOR THE SIMPLE COMPUTER (ED2900A)
      ;
      ;
      ;
        0000 START:  CODE CONT,      ,  DIS,  DTF,      , NRIR,  PCD,  FMAR    ; PC -> MAR
 0001         CODE CONT,      , READ,  RTF, MDOR,  RIR,      ,  FMAR    ; FETCH
 0002         CODE JOPC,      ,  DIS,      ,      , NRIR,      ,  NFD     ; DECODE STEP
 0003 LDA::   CODE JMP, FETCH, READ,  RTF, MDOR, NRIR,      ,  FACC    ; LOAD ACC
 0004 STO::   CODE JMP, FETCH, WRITE,     ,      , NRIR, ACCD,  NFD     ; STORE ACC
 0005 LADD::  CODE JMP, FETCH, READ,  ADD, MDOR, NRIR, ACCD,  FACC ; ACC + MEM
 0006 LSUB::  CODE JMP, FETCH, READ,  SUB, MDOR, NRIR, ACCD,  FACC    ; ACC - MEM
 0007 LOR::   CODE JMP, FETCH, READ,   OR, MDOR, NRIR, ACCD,  FACC    ; ACC OR MEM
 0008 LAND::  CODE JMP, FETCH, READ,  AND, MDOR, NRIR, ACCD,  FACC    ; ACC AND MEM
 0009 XOR::   CODE JMP, FETCH, READ, EXOR, MDOR, NRIR, ACCD,  FACC    ; ACC EXOR MEM
 000A INA::   CODE JMP, FETCH,  DIS,  RTF,  DIR, NRIR,      ,  FACC    ; DATA TO ACC
 000B OUT::   CODE JMP, FETCH,  DIS,      ,      , NRIR, ACCD,  NFD     ; ACC TO DATA-OUT
 000C GOTO::  CODE CONT,      ,  DIS,  DTF,      , NRIR,  PCD,  FMAR ; GO TO addr
 000D         CODE JMP, START, READ,  RTF, MDOR, NRIR,      ,  FPC     ; REFETCH INSTR
 000E IF::    CODE JIFO, GOTO,  DIS,      ,      , NRIR,      ,  NFD     ; IF ACC = 0 GO TO addr
 000F FETCH:  CODE JMP, START,  DIS,  ONE,      , NRIR,  PCD,  FPC     ; PC <- PC + 1
      ;
      ;
      END
```

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.0
SAMPLE .SRC FILE FOR THE SIMPLE COMPUTER (ED2900A)

 0000 00XXXXXXXXXXXX10 111X1110
 0001 00XXXXXXXXXXXX00 11010X10
 0002 11XXXXXXXXXXXX10 XXXX1X11
 0003 0100000000111100 11011X00
 0004 0100000000111101 XXXX1011
 0005 0100000000111100 00011000
 0006 0100000000111100 00111000
 0007 0100000000111100 01111000
 0008 0100000000111100 01011000
 0009 0100000000111100 10011000
 000A 0100000000111110 11001X00
 000B 0100000000111110 XXXX1011
 000C 00XXXXXXXXXXXX10 111X1110
 000D 0100000000000000 11011X01
 000E 1000000000110010 XXXX1X11
 000F 0100000000000010 101X1101



SAMPLE .SRC FILE FOR THE SIMPLE COMPUTER (ED2900A)

 ENTRY POINTS

  GOTO      000C
  IF        000E
  INA       000A
  LADD      0005
  LAND      0008
  LDA       0003
  LOR       0007
  LSUB      0006
  OUT       000B
  STO       0004
  XOR       0009
```

```
TITLE MEMORY MAP PROGRAM FOR THE "VERY SIMPLE COMPUTER"
;
;
WIDTH 8
;
; ASSUME A 32 x 8 PROM
;
BASE 16
;
  LDA
  STO
  LADD
  LSUB
  LOR
  LAND
  XOR
  INA
  OUT
  GOTO
;
END
```

.OPC

AMMAP

```
AMDOS/29 AMMAP ASSEMBLER, V1.1
MEMORY MAP PROGRAM FOR THE "VERY SIMPLE COMPUTER"

      ;
      ;
      WIDTH 8
      ;
      ; ASSUME A 32 x 8 PROM
      ;
      BASE 16
      ;
 0000   LDA
 0001   STO
 0002   LADD
 0003   LSUB
 0004   LOR
 0005   LAND
 0006   XOR
 0007   INA
 0008   OUT
 0009   GOTO
      ;
      END
```

```
MEMORY MAP PROGRAM FOR THE "VERY SIMPLE COMPUTER"

 0000 00000011 (0003)
 0001 00000100 (0004)
 0002 00000101 (0005)
 0003 00000110 (0006)
 0004 00000111 (0007)
 0005 00001000 (0008)
 0006 00001001 (0009)
 0007 00001010 (000A)
 0008 00001011 (000B)
 0009 00001100 (000C)
```

```
AMDOS/29 AMMAP ASSEMBLER, V1.1
MEMORY MAP PROGRAM FOR THE "VERY SIMPLE COMPUTER"

 ENTRY POINT SYMBOLS

  GOTO      000C
  IF        000E
  INA       000A
  LADD      0005
  LAND      0008
  LDA       0003
  LOR       0007
  LSUB      0006
  OUT       000B
  STO       0004
  XOR       0009

 TOTAL ASSEMBLY ERRORS =    0
```

AMD AMPROM UTILITY

SAMPLE .SRC FILE FOR THE SIMPLE COMPUTER (ED2900A)

PROM MAP

AMPROM

	PC	C1	C2	C3
R1	0000	1	2	3
R2	0008	4	5	6

PROM CONTENTS

PC	ADD	P 1	P 2	P 3
0000	000	00000000	00000010	11101110
0001	001	00000000	00000000	11010010
0002	002	11000000	00000010	00001011
0003	003	01000000	00111100	11011000
0004	004	01000000	00111101	00001011
0005	005	01000000	00111100	00011000
0006	006	01000000	00111100	00111000
0007	007	01000000	00111100	01111000

PC	ADD	P 4	P 5	P 6
0008	000	01000000	00111100	01011000
0009	001	01000000	00111100	10011000
000A	002	01000000	00111110	11001000
000B	003	01000000	00111110	00001011
000C	004	00000000	00000010	11101110
000D	005	01000000	00000000	11011001
000E	006	10000000	00110010	00001011
000F	007	01000000	00000010	10101101