AMD CUSTOMER EDUCATION

THE 2900 FAMILY STUDY GUIDE AND TEACHER'S MANUAL

THE 2900 FAMILY

STUDY GUIDE AND TEACHER'S MANUAL

by

Donnamaie E. White, Ph.D.

ADVANCED MICRO DEVICES, INC.

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

P R E F A C E

This book was prepared with emphasis on the 2900 Family members in which there is the most interest and the most need for instruction. A few of the older family members have been included as a reference point for those designers or students who have previously been exposed to bit-slice design and microprogramming. Later editions will add the newer family members as they are released.

It is necessary that the reader have the latest edition of the **2900 Family Data Book** and it is suggested that a copy of the nine part series of application notes,

HOTBACs (How To Build A Computer)

(which is also to be published by McGraw-Hill, New York (1980)), be available for reference. For the beginner, obtain a copy of

White, D. E. Bit Slice Design:Controllers and ALUs

to be published by Garland, New York (1980).

As with the customer education seminars, the reader is assumed to be either an experienced SSI/MSI designer or an experienced assembly level programmer.

The first section of this study guide is a sample exam which essentially surveys how much you have picked up from: 1) reading the application notes and the data book; or 2) completing the Advanced Micro Devices Customer Education seminar, ED2900A, "Introduction to Designing with the 2900 Family". It is the actual exam given at the end of the customer seminar. The emphasis of the exam is on the Am2910 microprogram controller and on the new Am29203 RALU, with a few questions on related family members such as the Am2914 vectored priority interrupt controller, the Am2909/2911 microprogram sequencers and the Am2901 RALU.

The second section is a series of exercises grouped on specific parts. Not all members of the 2900 Family are included. Some will be added in later editions, as they are developed; some are not included because they are difficult to comprehend. The Am29116 bipolar microprocessor is included on an introductory level.

If you are just beginning, work through the starred exercise sets only. (You will have to change Am27S27 questions to run off of an Am2910.) These are the newer parts and any new designs should be done with these family members. Notice that no alpha designation follows the part number (such as Am2901C, Am2910A, etc.). Always use the latest Data sheet. Do not waste time on outdated data. The Am2909 exercises are included for those who have a given system to maintain or to enhance.

Section three is still evolving and consists of problems that have been used in ED2900A as homework design problems. (We had to have the classic traffic light and coffee machine controller problems!) These conceptually simple problems actually demonstrate good microcoding techniques for the Am2910. A beginner should look these problems over carefully. Structured Microprogramming (clean, readable code) is the objective. Always!

Section four is the set of chapter end exercises developed for the Garland text. Some of the exercises come from the ED2900A and the ED2900B Microprogrammable Computer Architecture seminars. The exercises are paced to the text.

Sections five, six and seven are the answers for sections one and two, and for parts of section four. Section three stands by itself. Every attempt was made to produce good microcode format. The microprogramming solutions are written using pseudo-mnemonic notation, some of which refers to the data sheets of the relevant parts and some of which was developed specifically for the exercises. The translations are intended to be obvious. For example:

FQ/2->RAMQ means a double precision down shift with a RAM and Q register, the Q register being the least significant half of the pair.

RAM3TORAM0 means to link the RAM shift register to itself such that the RAM3 pin of the most significant slice (MSS) is output into the RAM0 pin of the least significant slice (LSS).

SPF2 means special function 2(HEX) for the Am2903/29203.

Design of "typical" CPUs are given in abundance in the various chapters of HOTBACs and the 2900 Family Data book and such designs and microcode are not reproduced here. Section four questions which are not answered herein may be found in the references. Chapter 8 of the HOTBACs, for example, documents a minicomputer and gives 256 microinstructions for the design, including initialize and examples of the various addressing modes allowed.

H O W   D O   Y O U   B E G I N ?

First, what is it that you wish to design?

A high speed controller would use Am2910, Am29116. It may also use an Am2904. Adding interrupts might use one or more Am2914s.

A high speed CPU or an emulation would use Am2910, Am29203 (breadboard with the Am2903 until the Am29203 is available), and the Am2904. With interrupts, add the Am2914s.

A controller (computer or otherwise) requiring more than 4K microwords might replace the Am2910 with several Am2909/11s and an Am29811. A minicomputer instructions set would fit comfortably into 4K; a sophisticated high-speed hard disk controller might not.

A computer with a program control unit (versus one using the data ALU to perform addressing) would build the PCU from Am2901s or Am2930/32s.

Second, study the data sheets for the parts in which you have an interest, one at a time. Try to answer the questions for those parts that are included in section two.

Third, study the example problems in section three and retry questions that you missed.

Forth, read through the Garland text and the HOTBACs series (they are not identical!). Then, try to answer any remaining questions that you missed. You should at this point try the exam - just skip over the questions on the parts in which you have no interest.

Finally, study the answers and compare to your own.

W H A T   O T H E R   H E L P   I S   T H E R E ?

The AMD 2900 Family literature is available from Advanced Micro Devices from the literature distribution center or from the Bipolar Applications group. Your local distributor or sales office would also have a supply. Page 3-1 in the 2900 Family Data Book lists the application notes that are available.

The Customer Education Center at Advanced Micro Devices currently offers a three course series of seminars and workshops.

ED2900A Introduction to Designing With the 2900 Family

EDSYS29 Introduction to Designing With the AMC SYS/29 Development System (a development system for microprogram development and prototype check-out)

ED2900B Microprogrammable Computer Architecture (everything you need to consider to develop a new CPU or emulate an old one with the 2900 Family)

Additional seminars are being developed on topics such as high speed controller design (the Am29116) and computer arithmetic (Am29203/2903 algorithms).

ED2910, "Basics of Bit-Slice", is a new course to be offered this spring. It is more elementary than the very intense ED2900A and is specifically for the beginner. If you have never microprogrammed, don't really know what that is, have never written or read assembly level programs, then this is the class that we recommend.

Call us at: 408-732-2400 ext 3665 or ext 2325
Outside California: 800-538-8450 ext 3665 or ext 2325

Write us at:
Advanced Micro Devices, Inc.
Customer Education Center
490-A Lakeside Drive
P.O. Box 435
Sunnyvale, CA 94086

[illegible]

Donnamaie E. White, Ph.D.

* marks more important parts

SAMPLE EXAM

The following is a sample exam for ED2900A or any introductory course on the 2900 Family of Advanced Micro Devices.

1. Show the microcode (partial width only) to program these statements, assuming an Am29803-Am2909/11-Am29811 CCU.

```
        IF A THEN ON (T2T0) GO TO (10, 200, 30, 40)
             ELSE ON (T3T1) GO TO (20, 200, 10, 20)
        IF B THEN ON (T3T2T1) GO TO (10, 20, 30, 40,..)
             ELSE ON (T2T1T0) GO TO (100, 200, 300,...)
```

Where:

A and B are condition multiplexer input lines
T3, T2, T1, T0 are test inputs to the Am29803
10, 20, 200, etc. are labels of statements
The same label means the same statement
The statements may be considered to be the beginning of a microroutine of unknown length

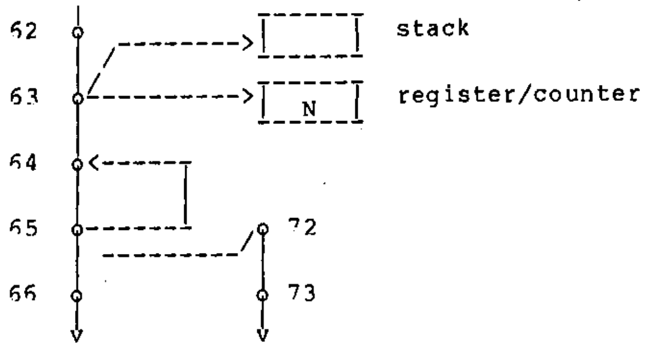
2. Show the microcode for the Am2910 instruction field and any other necessary sequence control fields to perform the following:

```
        IF A THEN DO S1
                     S2
                     S3
                  6 TIMES;
             ELSE DO S1
                     S2
                     S3
                  10 TIMES;
```

Where:

S1, S2, S3 are statement labels (the statements can do anything, we don't care)
A is an input to the conditional multiplexer

Given the following flow diagram:



3. What Am2910 instruction would be at address 63? (Show 3 sequence control fields.)

4. What Am2910 instruction would be at address 64?

5. What Am2910 instruction would be at address 65?

6. What value is on the stack (at the top) after executing the microinstruction at address 63?

7. Why use a pipeline register at the microprogram memory output instead of at the Am2910 address output lines?

8. How long does $\overline{CCEN}$ have to stay high to be detected?

9. List three ways to generate $\overline{OE}_{PL}$, $\overline{OE}_{VECT}$, $\overline{OE}_{MAP}$.

____
____
____

10. List two ways to "force a PASS" or TRUE at $\overline{CC}$ (Am2910); one way for a "PASS" at TEST (Am29811).

____
____

11. How much RAM is on-board the Am2901?

12. Is the RAM on the Am2901 expandable? If yes, how?

13. How much RAM is on-board the Am2903?

14. Is the RAM on the Am2903 expandable? If yes, how?

15. How much RAM is on-board the Am29203?

16. Is the RAM on the Am29203 expandable? If yes, how?

17. How are race conditions avoided in the Am2901?

18. Can the A, B addresses into the RAM on-board the Am2903 be identical, ie., may both address the same register in any one microcycle?

19. If the RAM register enable is enabled, when is data written into the selected register? (Any of the RALUs)

20. What is the guaranteed maximum capacitive load that the 2900 Family parts can drive without suffering some speed degradation?

21. Are the Am2903 and the Am29203 pin for pin compatible?

22. Which RALU supports BCD operation?

23. Which RALU supports byte operations best?

TRUE or FALSE

1. The Am2909/2911 and a PROM can form a flexible control unit

2. There is only one possible instruction set for the Am2910 if $\overline{RLD}$, $\overline{CCEN}$, and Ci are all held constant.

3. $\overline{RLD}$ on the Am2910 can be tied to instruction line I_1 or I_0 without harming the DO-loop instructions.

4. CJP, CJV, CJS, CJPP, JRP, and JSRP are all conditional branch statements in the Am2910 instruction set.

5. A shift and an arithmetic operation may be done in the same microcycle when using the Am2903.

6. The Am2903 is faster than the Am2901B on the average.

7. The functions available on the Am2901 are all available on the Am2903.

8. It is practical to emulate the Am9080 (In8080) using the 2900 Family.

9. The 2900 Family can be used to emulate the AmZ8000.

10. The 2900 Family currently supports real-time interrupts easily ("easily" refers to both implementation and conceptual clarity).

11. The Am2914 can be used for software level interrupt routines.

12. The Am2914 can be used for firmware level interrupt routines.

13. The Am2910 is a bit-slice device (expandable).

14. The 2900 Family is a bipolar technology family, TTL compatible.

15. The 2900 Family can be configured to operate with a 125ns microcyle via pipeline registers.

16. A disk controller would never use an ALU.

17. The Am2910 and Am2914 can be configured to handle subroutine or nonsubroutine calls to interrupt routines.

18. The Am2909 and the Am2910 have wraparound stacks.

19. The Am29803 attaches to the Am2911 to provide hardware supported case statements (N-way branching).

20. The Am29811 and Am2910 support the loop structures DO While X is FALSE (DO UNTIL X is TRUE) and DO i TIMES.

21. The Ri and Di inputs are tied together on the Am2909 and the Am2910.

EXERCISES ON SPECIFIC PARTS

EXERCISES - Am29811 - Am2909/2911

1. Diagram the pin-to-pin connection of an Am29811 and three Am2911s.

2. What is the logical value of $\overline{OE}$ if the Am29811 is to be always active?

3. Name an application where you would wish $\overline{OE}$ to be variable.

4. What are the counter load enable lines for?

5. Diagram the connection of an Am29811 and an Am25LS2569s (counters) for a 12-bit control memory address system.

6. Is the TEST input active-high or active-low?

7. Microcode the Am29811 to do the following:
(show: address Am29811 instr   TEST   Br Addr)

```
      10  *
          |
      11  *                          SUBROUTINE:
          |
 CJP  12  *---------->* 40             60 *
          |           |                   |
      13  *    JSUB   * 41             61 *
          |           |                   |
      14  *<--|       * 42      CJP    62 *-------->* 80
          |   |                           |         |
      15  *   |                        63 *<--------|
          |   | 10 TIMES                  |
      16  *-->|                 RET    64 *
          |
      17  *
```

EXERCISES - Am29803

1. How is the Am29803 turned off?

2. How are $\overline{OE}_1$ and $\overline{OE}_2$ used?

3. In any given design, how many different sets of 4 tests are possible without adding MUXs or other extra hardware (over and above the Am29803s)?

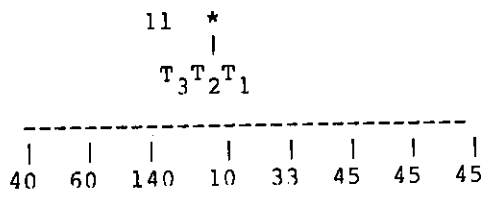
4. What instruction allows T_3 T_1 T_0 to be chosen for an 8-way branch?

T_3 T_0?

T_2 T_1?

T_3 T_2 T_1 T_0?

5. Microcode the Am29811 and Am29803 to do the following branch table: (show addr Am29811 instr Am29803 instr and the branch address)



6. Make a logic-level diagram of the Am29811 - Am29803 - Am2911/2909 configuration required to implement problem 5. above. Show PROM connections.

EXERCISES - Am27S27

Assume a 2Kx64 bit wide PROM memory has been designed using the Am27S27, and that Am2911s control the address lines.

1. Compute the address line current loads (worst case of course).

2. Are buffers needed?

3. What is the capacitive loading on the address lines?

4. What is the amount of speed degredation due to capacitive loading on the Am2911s? (Use 0.1ns/pf; 0.07ns/pf is typical.)

5. What can be done to the design to avoid the affect?

6. Repeat the loading computations for a 4Kx32 bit control memory.

7. Repeat the loading computations for a 4Kx64 bit control memory.

EXERCISES - Am2910

1. What does $\overline{RLD}$ = LOW do?

2. What does Ci = HIGH do?

3. What does $\overline{CCEN}$ do?

4. When should $\overline{FULL}$ be used?

5. Can two (2) Am2910s be hooked together in a system?

6. Name an application where you think you would want to do this.

7. Name two instances when you would want to vary $\overline{OE}$ into the Am2910.

8. How deep is the stack on the Am2910?

9. Is this the same as the Am2909/11 stack?

10. How does it differ if it does?

11. What happens if you PUSH 8 times and do not POP between them? (Use as many lines as are needed and diagram stack contents using 1,2,3,4,5,6,7,8.)

```
        -----------------
        -----------------
        -----------------
        -----------------
        -----------------
        -----------------
        -----------------
```

12. Microcode the Am2910 to do the following flow.
    (Show: address Am2910 instr Test Br addr)

```
        10  *
            |
        11  *                        SUBROUTINE:
            |
   CJP  12  *---------->* 40           60 *
            |           |                 |
        13  *     JSUB  * 41           61 *
            |           |                 |
        14  *<--|       * 42    CJP  62 *-------->* 80
            |   |                         |         |
        15  *   |                      63 *<--------|
            |   | 10 TIMES                |
        16  *-->|                RET  64 *
            |
        17  *
```

13. What are the main differences between the Am29811 and Am2910 instruction sets and control lines?

____
____
____
____

14. How does CJP differ from that of the 29811?

15. Can the Am2910 be used with an Am29803? If yes, how?

16. T or F: The 2910 is approximately equal to three 2911s, one 29811, one counter, and one decoder.

17. Explain why both RFCT and RPCT are needed.

18. Which is better if you are not sure about the reliability of the signal that you are using to terminate a DO-UNTIL loop: LOOP or TWB? And why?

19. Write the code for an exit from a subroutine if TESTA is TRUE, no test enable control.

20. Write the code for an $\underline{\text{unconditional}}$ exit from a subroutine.

21. Repeat problem 20 if $\overline{\text{CCEN}}$ is available.

22. Write a DO loop to execute one microinstruction 10 times, assuming that the stack is full and unavailable.

23. Repeat problem 22, but use the stack this time; $\overline{\text{CCEN}}$ is not available.

24. Repeat problem 23, but this time $\overline{\text{CCEN}}$ is available.

25. Write a DO loop to execute one microintruction until TESTB is true. Do $\underline{\text{NOT}}$ allow the register to load.

26. Write a DO loop to execute one microinstruction until TESTC is true OR the instruction has been executed 20 times. Branch to FAILADDR.
Write it $\underline{\text{without}}$ $\overline{\text{CCEN}}$ and then write it using $\overline{\text{CCEN}}$.

27. Write code to branch to ADR1 if TESTD is FALSE or to ADR2 if TESTD is TRUE. $\overline{\text{RLD}}$ and $\overline{\text{CCEN}}$ are not involved.

28. Write code to jump to the subroutine at SUBA if TESTD is FALSE or to the subroutine at SUBB if TESTD is true.

29. Write code to jump out of the middle of a DO loop (which uses LOOP, RFCT, or TWB) if TESTE is TRUE. Branch to address ADR3.

30. Using $\overline{RLD}$, load the register/counter with address SUBA three steps prior to the subroutine call above.

31. Draw the flow diagram for normal execution of the CJS instruction and show the sequence control portion of the microinstruction. CJS appears at address 13; the subroutine is at address 40; and TESTC is tested. (Yes, symbolic labels are best when writing microcode.)

32. Repeat 31, assume $\overline{CCEN}$ = HIGH.

33. Repeat 31, assume $\overline{RLD}$ = LOW.

34. Repeat 31, assume C_{in} goes LOW just after loading of the CJS instruction into the microinstruction register.

35. Devise a flowchart of the decision activity if TESTC, $\overline{RLD}$, $\overline{CCEN}$ and C_{in} are all present in the microword.

LABEL/ ADDR	2910 INSTR	COND MUX SEL	$\overline{CCEN}$	BR ADDR/ COUNTER

Am2910 CODING SHEET

LABEL/ ADDR	2910 INSTR	COND MUX SEL	$\overline{CCEN}$	BR ADDR/ COUNTER

EXERCISES - Am9140

1. How many address lines are available?

    ----

2. Are any enables needed?

    ----

3. Diagram a 4Kx32 bit memory.

4. If a 4Kx32 Am9124 memory is used for a control memory, what is the load on the Am2910 output lines (current and capacitance)?

    ----

5. Could an 8Kx32 bit memory be driven by an Am2910 (unaided)?

    ----

6. How would you do it?

EXERCISES - Am2914

1. How many levels of prioritized interrupt requests can be handled by one Am2914?

2. Are interrupt requests active high or active low?

3. If you wish to mask an interrupt level, do you set the mask bit to 1 or to 0?

4. Is this correct: $INR_7 \cdot \overline{MK}_7 = IR_7$
where INR_i is the complement of the actual interrupt request line, MK_i is the mask register bit, and IR_i is the result of the mask operation?

5. What Am2914 instruction causes the binary coded vector V_i to be loaded into the Vector Hold Register?

6. What is the function of the vector hold register?

7. When is the vector hold register cleared?

8. What does the binary coded vector represent?

9. Is the status register set equal to the priority of the interrupt currently being serviced?

10. If the status fence is 5, what non-masked interrupts may be recognized?

11. If four Am2914s are present in a system, how many lowest group enabled flip-flops can be low at a time (and still have correct operation)?

12. Correct? :

$$\overline{\text{INTERRUPT REQUEST}} = \overline{\text{DET } \overline{\text{INTERRUPT DISABLE}} \cdot \text{PASS PRIORITY}}$$

$$\overline{+ \text{ PASS ALL}}$$

EXERCISES - Am2901

1. Can the Q register be shifted by itself (without shifting the RAM shift register)?

2. Is it possible to perform R_c <-- R_b + R_a in one microcycle?

3. What is connected to Y_i during execution of a NOP?

4. Does the Am2901 perform a logical NAND or NOR function in one microcycle?

5. Is the Am2901 designed for one's or two's complement arithmetic?

6. Is it possible to perform D_a + D_b --> Y_i, where D_a and D_b are from outside the chip sources, in one microcycle?

7. Diagram the interconnection of a 16-bit ALU that can perform RAM and Q register up and down shifts and rotates. Use the Am2901, Am2902 and multiplexers.

8. Show the microword to do a single precision rotate of 1 bit down on register 6. (2910 instr., muxsel., braddr., ALU source, ALU func., ALU dest., A addr., B addr., Cin, rotate muxsel.)

9. Refer to problem 8. above but do a rotate of 2 bits up in one microcycle.

10. Is it possible to do R + C_{in} in one microcycle? Show the code.

11. Show the code required (for the 2910, 2901) to perform D_A + D_B --> Y_i.

12. Show the code to perform R_a + R_b --> R_c.

13. Show the code to perform a byte swap on R_6 assuming no hardware assist.

14. Show the code to perform PC --> MAR; PC <-- PC + 1, where PC = R_{15}.

Am2910 - Am2901 CODING SHEET

ADDR	2910 INST	MUX SEL	BR ADDR COUNTER	A SRCE	L FUNC	U DEST	RA ADDR	RB ADDR	Cin SEL	ROTATE MUX SEL
8.										
9.										
10.										
11.										
12.										
13.										
14.										

Which instruction lines provide or do the following?

Am2901	$I_8I_7I_6$	$I_5I_4I_3$	$I_2I_1I_0$
Select of MUX at R(ALU)			
Function select of ALU			
Enables the Q register			
Select of Output MUX			
Inhibit at R or S			
Select of MUX at RAM input			
Select of MUX at QREG input			
RAM enable			
RAM_0 enable			
RAM_3 enable			
Q_0 enable			
Q_3 enable			
Select of MUX at S(ALU)			

EXERCISES - Am2903 - Am29203

1. What is the difference between a logical and an arithmetic shift?

2. Can the RAM shift register perform both logical and arithmetic shifts?

3. Can the Q shift register perform both logical and arithmetic shifts?

4. Can the Q register be shifted independently of the RAM shift register?

5. How many special functions are implemented on the Am2903?

6. Is it possible to load the Q register and not load the RAM register?

7. Does the Am2903 perform the logical functions NAND and NOR in one microcycle?

8. Does the Am2903 perform $R_a + R_b \rightarrow R_c$ in one microcycle?

9. Does the Am2903 perform $D_a + D_b \rightarrow Y_i$ in one microcycle, where D_a and D_b come from outside the chip sources?

10. Can the Am2903 perform $S + C_{in}$, $R + C_{in}$, $S + C_{in}$ and $R + C_{in}$, each in one microcycle?

11. Which instruction lines control whether or not the special functions are active?

12. Can the Q register be shifted as it is initially loaded (F --> Q/2; F --> 2Q)?

13. Can any RAM register be loaded when $\bar{I}_{EN}$ is high?

14. Are the status lines affected if single length normalize is executed with $\bar{I}_{EN}$ = HIGH?

15. Show the code (2910, 2903) to perform $D_a + D_b \dashrightarrow Y_i$.

16. Show the code to perform $R_a + R_b$ --> R_c.

17. Show the code to perform a byte swap with no hardware assist.

18. Show the code perform PC --> MAR; PC <-- PC + 1, where PC = R_{15}.

19. Show the code to perform D_A --> Q.

20. Output <R_2> and perform 2*(R_2 + 1) in <u>one microcycle</u>.

21. Perform an unsigned 16-bit multiply.

22. Perform a two's complement 16-bit multiply.

23. Perform a double precision down shift using R_2 and Q.

24. Perform 4*R_2 --> Q.

Am2910 - AM2903/29203 CODING SHEET

LABEL/	2910	MUX	BRADDR	A	L	U	RA	RB	Cin	ROTATE
ADDR	INST	SEL	COUNTR	SRCE	FUNC	DEST	ADDR	ADDR	SEL	MUX SEL
15.										
16.										
17.										
18.										
19.										
20.										
21.										
22.										
23.										
24.										

Which group of instruction lines or enables controls the following?

Am2903	$\bar{E}_A I_0 \overline{OE}_B$	$I_4 I_3 I_2 I_1$	$I_8 I_7 I_6 I_5$
Input MUX at R			
RAM enable			
Switch to special functions			
ALU function			
WRITE enable			
Q_0 enable			
Q_3 enable			
SIO_0 enable			
SIO_3 enable			
Q Register enable			
Input MUX at S			
Chip function (during special functions)			
DB_{0-3} enable			
Status output select			
Inhibit at R and S			
Q input MUX			
RAM input MUX			

a. What four functions could appear at C_{n+4} under instruction control?

b. How many different functions could appear at the $\overline{P}$/OVR pin under instruction control?

c. List them.

d. How many different functions could appear at the $\overline{G}$/N pin under instruction control?

e. List them.

f. How many different functions could appear at the Z pin under instruction control?

g. List them.

h. T or F: $\overline{I}_{EN}$ controls $\overline{WRITE}$ on the Am2903 and Am29203.

i. Can special functions be executed when $\overline{I}_{EN}$ = LOW?

j. What happens?

k. If the scratchpad RAM is not expanded past 16 registers, what connects to $\overline{WE}$?

EXERCISES - Am29705/29707

1. How many address lines are required?

2. How many enables are there on this chip?

3. If Am29705s are attached to Am2903s, what is connected to the write enable lines?

4. Refer to 3. above, what connects to $\overline{LE}$?

5. If two Am29705s, one Am2903 are in a system, can R_C <-- R_A+ R_B have one address from each of the three devices?

6. Refer to 5. above, can data be brought directly from the data bus to any 29705 in one microcycle?

7. Refer to 5. above, how many address lines are needed if all three devices may be used in any one microinstruction? If only one may be used in any one microinstruction?

EXERCISES - Am2904

Using HEX code (mnemonics haven't yet been created), write code for the Am2904 and the destination field of the Am2901 to perform:

1. F/2 --> RAM, hold Q, ZERO --> $RAM_3|_{MSS}$

2. FQ/2 --> RAMQ, ZERO --> $RAM_3|_{MSS}$

3. 2FQ --> RAMQ, ONE --> $RAM_0|_{LSS}$

4. 2F --> RAM, $RAM_3|_{MSS}$ --> $RAM_0|_{LSS}$, hold Q

Using HEX code write code for the Am2904 and the destination field of the Am2903 to perform:

5. F/2 --> RAM, hold Q, M_N --> $SIO_3|_{MSS}$

6. FQ/2 --> RAMQ, M_N --> $SIO_3|_{MSS}$

7. F/2 --> RAM, hold Q, I_C --> $SIO_3|_{MSS}$

8. F/2 --> RAM (arithmetic shift), hold Q, $I_N \oplus I_{OVR}$ --> $SIO_3|_{MSS}$ (careful!)

9. What values on what instruction lines are necessary to cause C_{in} to the Am2901 or Am2903 to be set as μ_C ($C_o|_{2904} = \mu_C$)?

10. What values on what instruction lines cause the Machine Status Register, MSR, to be copied into the microstatus register, μSR, and MSR <-- Y_i?

11. How do you swap MSR and μSR?

12. How do you set μSR only (not MSR)?

13. How do you reset μSR?

14. How do you set the M_C bit only?

15. How do you reset the M_C bit only?

16. How do you swap M_C and M_{OVR}?

17. Write code for the 2904 to perform the conditional test $(\mu_N \oplus \mu_{ovr}) + \mu_Z$ and not alter the status register?

18. IF $\overline{CE}_\mu$, $\overline{CE}_M$, $\overline{E}_Z$, $\overline{E}_C$, $\overline{E}_N$, $\overline{E}_{OVR}$ are all LOW for the above, what happens?

19. Write code to test I_{OVR}; $\overline{CE}_\mu$, $\overline{CE}_M$ are HIGH.

20. Repeat 18. if CE_μ = LOW. What happens?

21. Write code to test M_C.

Show microcode for the Am2910, Am2901 and Am2904 to preform the following:

22. R_{15} --> Y; R_{15} + 1 --> R_{15}.

23. Byte swap register 6 using no external hardware.

24. Byte swap register 6 using SSI/MSI hardware assist.

25. What are the differences in the microwords for the above exercises and those written for the Am2901 exercises?

EXERCISES - Am29116

True or False:

1. The Am29116 is a TTL compatible, ECL internal device.

2. The Am29116 is expandable (two can be hooked together).

3. The Am29116 is for 8 bit or 16 bit intelligent controllers.

4. The Am29116 can perform conditional testing on its status register.

5. The barrel shifter 'shifts' (rotates) 1 to 15 bits up or down in one microcycle.

6. The Am29116 must be used with an Am2904.

7. The Am29116 can perform immediate operations.

8. The Am29116 has a choice of four input sources to the MUX which in turn provides three ALU inputs, R, S, U.

9. The Am29116 can perform three address instructions.

10. The Am2922 (MUX) and the Am2904 (GLUE) both have polarity control on the conditional inputs.

11. Fast clock speed is synonomous with high throughput.

12. The Am29116 can generate CRC polynomials up to 16 bits long.

13. The Am29116 always has its ALU output at Y_i.

14. The ALU destinations are RAM, ACC, D Latch.

15. Single operand instructions are PASS, COMPLEMENT, INCREMENT, and TWO'S COMPLEMENT.

16. D_{OE} (D with zero extend) is used for Two's complement arithmetic.

17. $\bar{R}$ --> DEST is One's complement, $\bar{R}$ + 1 --> DEST is Two's complement.

18. The Am29116 can perform NAND, NOR, EXOR in one microcycle.

19. Shift up can use 0, 1, or the QLINK bit as input to the LSB.

20. Shift down uses 0, 1, or the QLINK bit as the only input choices to the MSB.

21. Rotate can work in byte or word mode.

22. Rotate uses the U ALU input.

23. LOAD 2^n causes a mask (1 in a field of 0s) to be generated and used for loading RAM, ACC.

24. Read bus, change bit, output to bus is possible in one microcycle with the Am29116.

25. If you bit change the ACC, the destination is the ACC or the RAM.

26. There are 17 choices for priority encoding.

27. Byte mode prioritize does not use bits 8 - 15.

28. The Am29116 can perform operations on up to and including 16 bit CRC polynomials.

29. 95% of CRC calculations use 16 bit polynomials.

30. The CRC calculations can be done forward or reverse (transmit bit 0 first or transmit bit 15 first).

31. CRC can be done in byte or word mode.

32. The status word can be loaded from D, RAM, ACC.

33. The Z, C, N, OVR status bits can be loaded <u>without</u> affecting LINK or FLAGs.

34. You can set or reset the entire status word on the Am29116.

35. You can set or reset the arithmetic flags individually on the Am29116.

36. On the Am29116, you can load the status and the ACC registers both in the same microcycle if the RAM is the source.

37. If the status register enable is on, the status register is frozen (any operation on the status register is "killed").

38. All conditional testing is performed on the stored values in the status register.

FILL IN OR ANSWER:

39. How many bits in the Am29116 status register?

40. How many conditional tests can be made?

41. Can you perform a conditional test during another instruction? If so, how?

42. Can byte operations be performed in the upper or lower byte of the 16-bit Am29116?

43. Can the Am29116 support a 100ns microcycle time?

44. List three possible sources for single operand instructions.

45. Can you load the D (data) latch one byte at a time?

46. List three possible source pairs for two operand instructions.

47. If n = 2, show the pattern in the 16-bit word after a word rotate given:

```
1 1 1 1 1 1
5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
1 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1
```

48. Does the Am29116 support $<R_3>$ .ROTATE. --> $<R_3>$?

49. Does the Am29116 support $<R_3>$ .ROTATE. --> $<R_7>$?

50. If there is no priority request, what is produced by the priority encoder (word mode)?

51. If bit 15 is active, what is produced (word mode)?

52. If bit 15 is active, what is produced (byte mode)?

53. The Am29116 is an order of magnitude faster than the Am2901 which is an order of magnitude faster than the AmZ8000 for specific controller oriented operations. What can you say about program area?

54. Can the Am29116 be used to do multiply?

55. Can the Am29116 be used for bit operations?

56. Can the Am29116 be used for rotate operations?

57. Does the Am29116 have an ALU?

58. Can the Am29116 be used to build a CPU?

59. If the mask bit i is zero in a ROTATE-MERGE instruction, which operand's bit i is passed to the destination?

60. If
```
   U = 0011 0001 0101 0110
   R = 1010 1010 1010 1010
MASK = 0101 1010 0110 1001
```
and n = 4
what bit pattern is produced by a word mode ROTATE-MERGE instruction?

61. If the highest bit position with a one (1) is bit position 7, and the mask is 1010_{HEX}, what is produced by a word prioritize instruction?

62. Repeat for byte mode prioritize.

63. For LOAD 2^N complement, ($\bar{2}^N$), what happens?

64. Suppose you want to do a word rotate down five bit positions. How do you do this on an Am29116?

65. What is QLINK?

66. Can you set/reset the ALU status bits one by one as with the Am2904 Machine status register?

67. Can you set/reset the LINK and FLAGi status bits one at a time?

68. In byte mode, are the lower 8 bits of the status register loaded?

69. Which instructions do **NOT** cause the ALU status bits to be updated?

70. When are the upper four status bits (LINK, FLAG1, FLAG2, FLAG3) changed?

DESIGN PROBLEM :

TRAFFIC LIGHTS

Assume a simple 4-way intersection, no protected left turns, nothing fancy required of the controls except that one street is RED while the other shows GREEN and YELLOW occurs between the RED and GREEN lights and visa versa (for simplicity).

You have access to a 10 second clock signal.

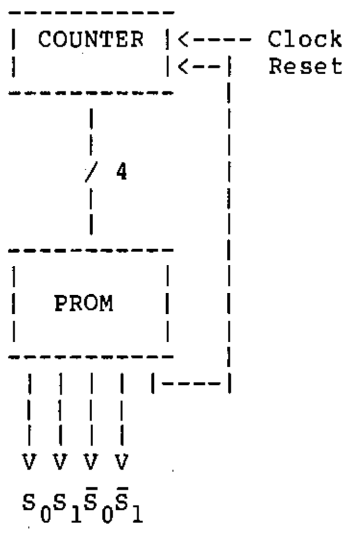
Timing is:

RED-GREEN	60 seconds
YELLOW	10 seconds

Controls on all traffic light units are identical (same decoding).

Use a counter and a PROM and any SSI you need and design a traffic light sequence controller for this intersection.

(And, if you are a masochist, try doing this with D F/Fs or even J/Ks!)

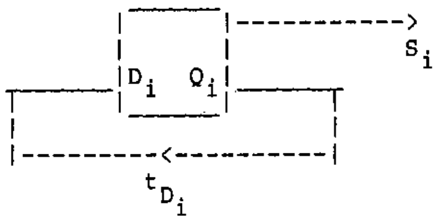


MICROPROGRAM

	ACTUAL PROM CONTENTS					PSEUDO-ASSEMBLY MICROPROGRAM			
ADDR	S_0	S_1	$\bar{S}_0$	$\bar{S}_1$	RESET	LIGHT 1	LIGHT 2	RESET CNTRL	
0	0	0	1	1	0	RED	GREEN	N	
1	0	0	1	1	0	RED	GREEN	N	
2	0	0	1	1	0				
3	0	0	1	1	0	etc.			
4	0	0	1	1	0				
5	0	0	1	1	0	RED	GREEN	N	
6	0	1	1	0	0	YELLOW	YELLOW	N	
7	1	1	0	0	0	GREEN	RED	N	
8	1	1	0	0	0				
9	1	1	0	0	0	etc.			
10	1	1	0	0	0				
11	1	1	0	0	0				
12	1	1	0	0	0	GREEN	RED	N	
13	1	0	0	1	1	YELLOW	YELLOW	Y	<--- RESET FOR COUNTER

SEQUENCE (GREY CODE)

Q_3	Q_2	Q_1	Q_0	S_1	S_0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	1	0	0
0	0	1	0	0	0
0	1	1	0	0	0
0	1	1	1	0	1
0	1	0	1	1	1
0	1	0	0	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	1	1	1
1	0	1	1	1	1
1	0	1	0	1	0
1	0	0	0	0	0



EQUATIONS

$D_0 = \bar{Q}_3\bar{Q}_2\bar{Q}_1 + Q_2Q_1 + Q_3Q_2$ <---- NEXT STATE CONTROLS

$D_1 = \bar{Q}_3Q_1\bar{Q}_0 + \bar{Q}_2Q_0 + Q_3Q_0$

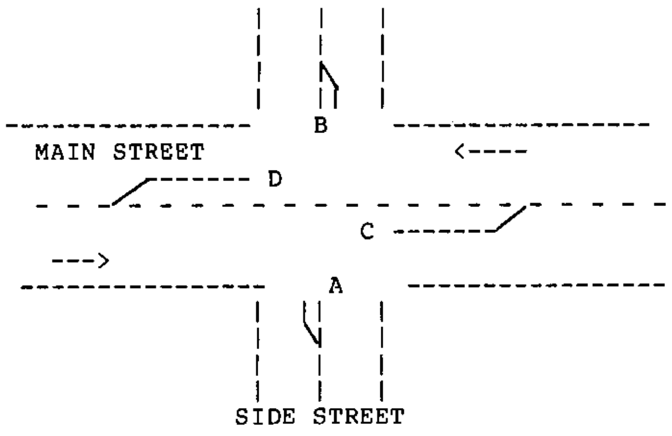
$D_2 = \bar{Q}_3Q_1\bar{Q}_0 + Q_2\bar{Q}_1 + \bar{Q}_3Q_2$

$D_3 = Q_2\bar{Q}_1\bar{Q}_0 + Q_3Q_2 + Q_3Q_1$

$S_1 = Q_2\bar{Q}_{\underline{1}} + Q_3Q_0 + Q_3Q_1$ <---- LIGHT CONTROL

$S_0 = Q_2Q_{\overline{1}} + Q_3Q_0 + Q_2Q_0$

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	10s	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.

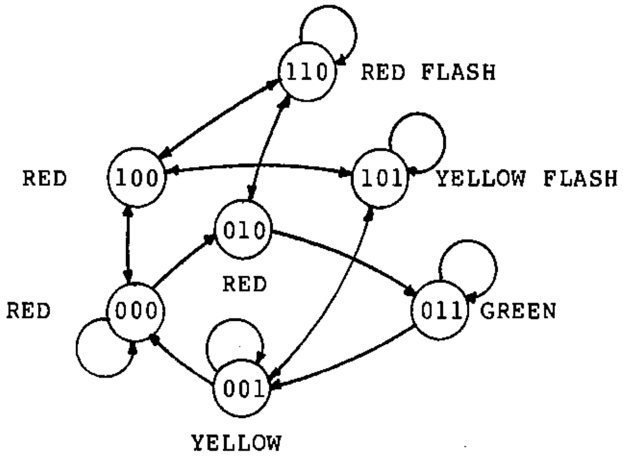
Five states means three lines of encoded controls into the individual traffic lights:

$$I_2 I_1 I_0$$

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., <u>only one signal line may change per clock step per light</u>.

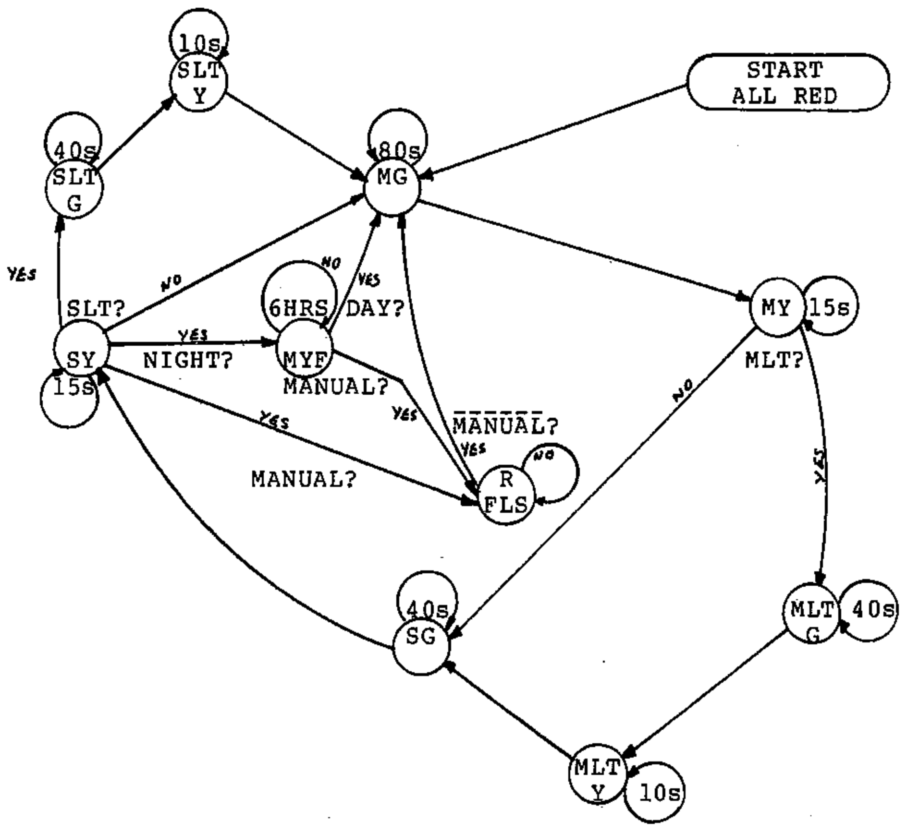
SAMPLE SEQUENCE

$I_2 I_1 I_0$	
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

```
                                ------------
                                |          |
           MUX         Am2910   |          |
           ----            -----V----<----------CLOCK
MLT ------>|3 |            |    Di  |      |
SLT ----->o|2 |            |        |      |
           |  |----------->|CC      |      |
NIGHT ---->|1 |            |        |      |
MANUAL---->|0 |      ----->|OE      |      |
           ----      |  -->|Ii      |      |
            ↑SEL.    |  |  ----------      |
START ----------------  |       |          |
            |      ------       / 6        / 6
            |      |            |          |
            / 2    |   ---------V--------  |
            |      / 4 |                |  |
            |      |   |     PROM       |  |
            |      |   |                |  |
            |      |   ------------------  |
            |      |   | PIPELINE REG.  |<----- CLOCK
            |      |   ------------------  |
            |      |    |   |   |   |      |
            |      -<----   |   |   ------>-
            -<---------------   |
                                / 12
                                |
                                V  LIGHT CONTROL
                                   SIGNALS
```

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	$\overline{\text{SLT}}$?	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.

DESIGN PROBLEM :

THE FAMOUS COFFEE MACHINE

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 dispenced.

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.

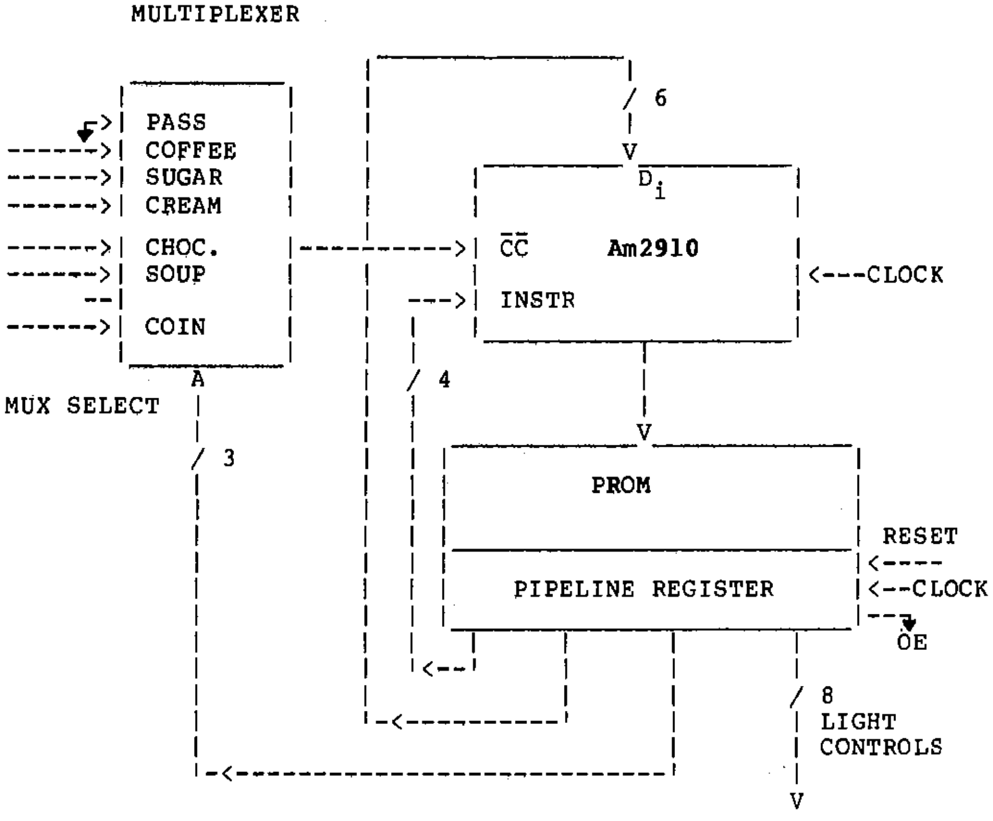
sugar	1.5 seconds
cream	2.0 seconds

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

HARDWARE CONFIGURATION



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	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

SIMPLE PROBLEMS FOR BEGINNERS
THE FAMOUS COFFEE MACHINE

```
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
-------------------------------------------------------------
```

DESIGN PROBLEM :

A VERY SIMPLE COMPUTER

Assume that you have a simple computer such as is shown in the figure on the following page. It consists of an elementary control unit which receives as input:

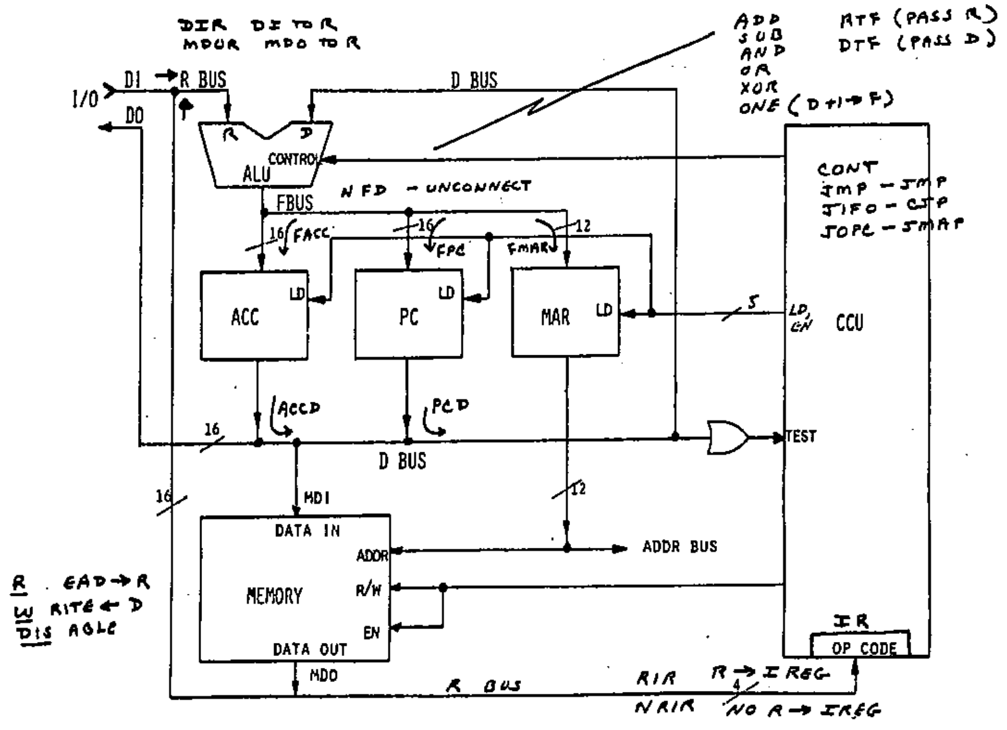
- the op code from the instruction
- the result of a test on the contents of the ACC
  - <ACC> = 0

and which generates as output:

- the function select control for the ALU
- the carry-in for the ALU
- the load and enable controls for the
  - MAR memory address register
  - PC program counter
  - ACC accumulator
- the read/write controls for the memory
- the enable for the memory
- controls for all of the bus signals

the R bus	data-in bus; also to ALU R port
the D bus	data-out bus; also to ALU D port
the F bus	accumulator output

HARDWARE CONFIGURATION AND CONTROLS



— GIVEN —

BASIC COMPUTER INSTRUCTION SET

The following assembly-level single address instructions are to be supported by the hardware through microprogramming. You are to assign the opcode and the starting address for the supporting microroutine (assume a memory map).

SYNTAX	SEMANTICS	
LDA, addr	<ACC> <-- <addr>	
STO, addr	<addr> <-- <ACC>	
ADD, addr	<ACC> <-- <ACC> + <addr>	
SUB, addr	<ACC> <-- <addr> - <ACC>	
OR, addr	<ACC> <-- <ACC> .OR. <addr>	
AND, addr	<ACC> <-- <ACC> .AND. <addr>	
XOR, addr	<ACC> <-- <ACC> .EXOR. <addr>	
INA	<ACC> <-- <DI bus>	
OUT	<DO bus> <-- <ACC>	
JMP, addr	<PC> <-- addr	jump immediate
JMP, addr	<PC> <-- <addr>	jump indirect
JMZ, addr	IF <ACC> = 0 THEN <PC> <-- addr	immediate
JMZ, addr	IF <ACC> = 0 THEN <PC> <-- <addr>	indirect

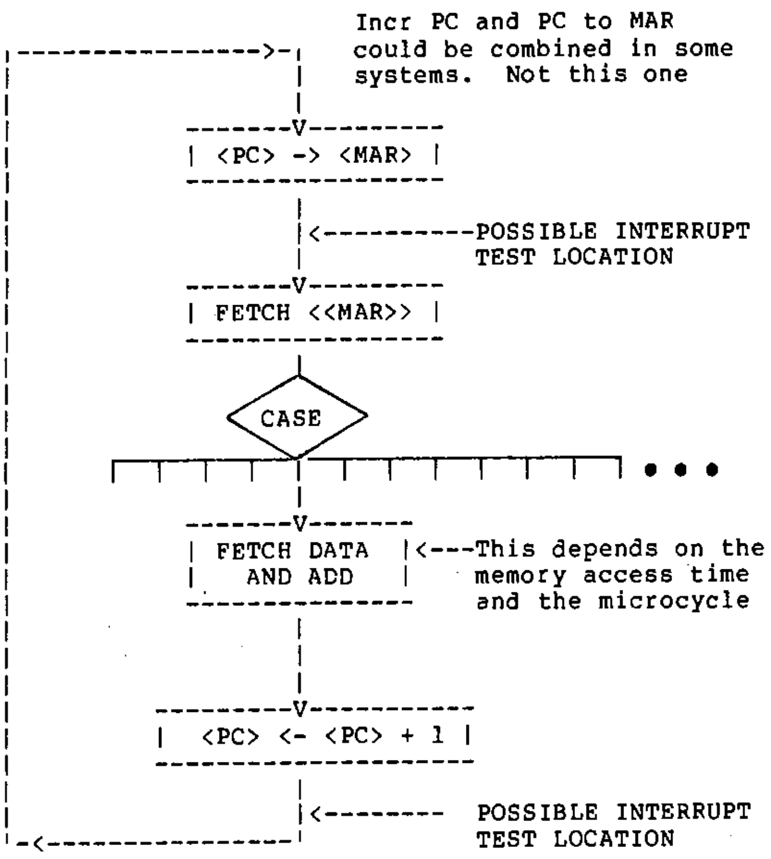
You may choose to do either immediate or indirect jump instructions (not both)

The <u>entire microprogram</u> should not exceed 16 microinstructions

MICROROUTINE FOR THE ADD INSTRUCTION

```
START:
<PC> --> <MAR>                 Contents of PC register to MAR
<<MAR>> --> <IR>, <MAR>        Fetch instruc. to IR (opcode)
                               and MAR (addr)
GOTO ADD                       Case jump on opcode
<ACC> <-- <ACC> + <<MAR>>      Fetch data and add
<PC> <-- <PC> + 1              Increment PC register
GOTO START
```

This is a conventional flow for a CPU


Incr PC and PC to MAR could be combined in some systems. Not this one
<PC> -> <MAR>
POSSIBLE INTERRUPT TEST LOCATION
FETCH <<MAR>>
CASE
FETCH DATA AND ADD
This depends on the memory access time and the microcycle
<PC> <- <PC> + 1
POSSIBLE INTERRUPT TEST LOCATION

The following is one solution for the simple computer design.

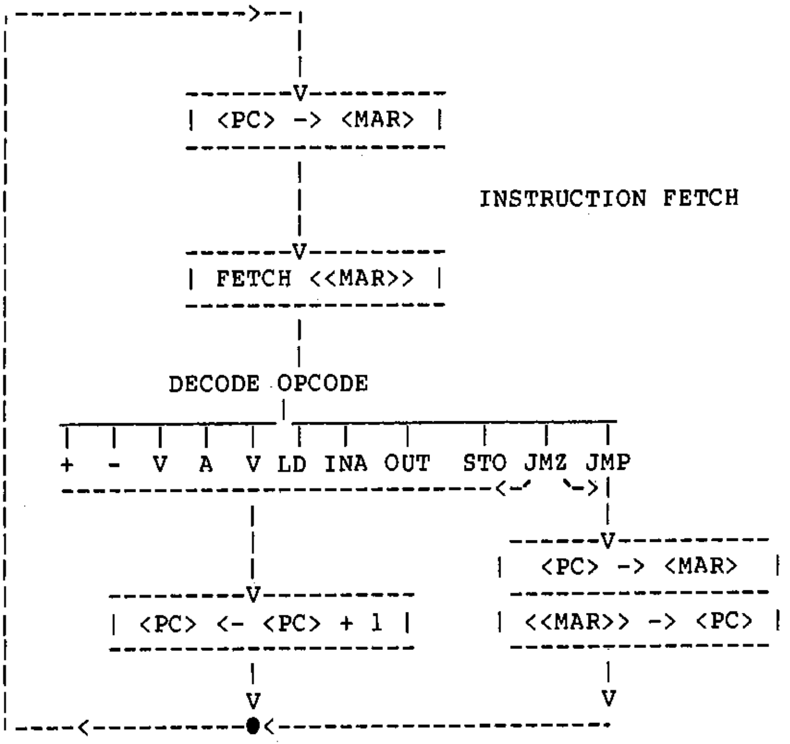
Step one: define all of the controls

NEXT ADDR INSTR:	CONT
	JMP
	JIF0 (CJP)
	JOPC (JMAP)
ALU FUNCTIONS:	ADD
	SUB
	AND
	OR
	XOR
	ONE (D+1 --> F)
	RFT (PASS R)
	DTF (PASS D)
R BUS CONTROL:	DIR (D-IN TO R-PORT)
	MDOR (MEMORY TO R-PORT)
F BUS CONTROL:	FACC (F TO ACC)
	FPC (F TO PC)
	FMAR (F TO MAR)
	NFD (NO CONNECTION)
D BUS CONTROL:	ACCD (ACC TO D)
	PCD (PC TO D)
MEMORY CONTROL:	READ
	WRITE
	DISABLE
INSTR REG CONTROL:	RIR (READ OPCODE TO IR)
	NRIR (NO IR CONNECTION)

Step two: design first-draft of microword format

ADDR LABEL	NEXT ADDR	BRANCH ADDR	MEMORY CONTRL	ALU SEL	SOURCE R BUS	IR CONTRL	D BUS	F BUS
(BITS)	2	4	2	3	1	1	2	4

Step three: develop a flowchart of the microprogram to see where the microroutines can share microinstructions. Even a crude worksheet is important documentation. The more comments on it, the easier it is for a stranger to follow.


<PC> -> <MAR>
FETCH <<MAR>>
INSTRUCTION FETCH
DECODE OPCODE
+
-
V
A
V
LD
INA
OUT
STO
JMZ
JMP
<PC> <- <PC> + 1
<PC> -> <MAR>
<<MAR>> -> <PC>

Step 4: draft the microprogram from the definition file and the flowchart.

		NEXT	BRANCH	MEMORY	ALU	SOURCE	IR			
ADDR	LABEL	ADDR	ADDR	CONTRL	SEL	R BUS	CONTRL	D BUS	F BUS	COMMENT
0	START	CONT	#	DIS	DTF	#	NRIR	PCD	FMAR	PC->MAR
1		CONT	#	READ	RTF	MDOR	RIR	#	FMAR	FETCH
2		JOPC	#	DIS	#	#	NRIR	#	NFD	DECODE
3	LDA	JMP	FETCH	READ	RTF	MDOR	NRIR	#	FACC	
4	STO	JMP	FETCH	WRITE	#	#	NRIR	ACCD	NFD	
5	ADD	JMP	FETCH	READ	ADD	MDOR	NRIR	ACCD	FACC	
6	SUB	JMP	FETCH	READ	SUB	MDOR	NRIR	ACCD	FACC	
7	OR	JMP	FETCH	READ	OR	MDOR	NRIR	ACCD	FACC	
8	AND	JMP	FETCH	READ	AND	MDOR	NRIR	ACCD	FACC	
9	XOR	JMP	FETCH	READ	XOR	MDOR	NRIR	ACCD	FACC	
10	INA	JMP	FETCH	DIS	RTF	DIR	NRIR	#	FACC	
11	OUT	JMP	FETCH	DIS	#	#	NRIR	ACCD	NFD	
12	GOTO	CONT	#	DIS	DTF	#	NRIR	PCD	FMAR	
13		JMP	START	READ	RTF	MDOR	NRIR	#	FPC	REFETCH
14	IF	JIF0	GOTO	DIS	#	#	NRIR	#	NFD	
15	FETCH	JMP	START	DIS	ONE	#	NRIR	PCD	FPC	INCR PC

In this program jump direct to address is microprogrammed. Since there is no path from MAR to PC, the instruction must be <u>refetched</u>! ie., the hardware chosen has a more obvious effect on the actual microprogram than it has on the higher level language. The assembly level or the machine level programmer only needs to know the semantics and the syntax of the language.

Step 5: once the microprogram has been written it is possible to establish the memory map which will translate each opcode into the corresponding starting address for its microroutine.

INSTRUCTION	OPCODE		ROUTINE ADDRESS	
DESCRIPTION	NAME	#	LABEL	ACTUAL
LOAD ACC	LDA	0	LDA	3
STORE ACC	STO	1	STO	4
ADD ACC	ADD	2	ADD	5
SUB ACC	SUB	3	SUB	6
OR ACC	OR	4	OR	7
AND ACC	AND	5	AND	8
XOR ACC	XOR	6	XOR	9
INPUT ACC	INA	7	INA	A
OUTPUT ACC	OUT	8	OUT	B
JUMP addr	JMP	9	GOTO	C
JMP addr IF	JMZ	A	IF	D

The actual opcode assignment is unimportant in this case.

INDIRECT JUMP SOLUTION

ADDR	LABEL	NEXT ADDR	BRANCH ADDR	MEMORY CONTRL	ALU SEL	SOURCE R BUS	IR CONTRL	D BUS	F BUS	COMMENT
0	START	CONT	#	DIS	DTF	#	NRIR	PCD	FMAR	PC->MAR
1		CONT	#	READ	RTF	MDOR	RIR	#	FMAR	FETCH
2		JOPC	#	DIS	#	#	NRIR	#	NFD	DECODE
3	LDA	JMP	FETCH	READ	RTF	MDOR	NRIR	#	FACC	
4	STO	JMP	FETCH	WRITE	#	#	NRIR	ACCD	NFD	
5	ADD	JMP	FETCH	READ	ADD	MDOR	NRIR	ACCD	FACC	
6	SUB	JMP	FETCH	READ	SUB	MDOR	NRIR	ACCD	FACC	
7	OR	JMP	FETCH	READ	OR	MDOR	NRIR	ACCD	FACC	
8	AND	JMP	FETCH	READ	AND	MDOR	NRIR	ACCD	FACC	
9	XOR	JMP	FETCH	READ	XOR	MDOR	NRIR	ACCD	FACC	
10	INA	JMP	FETCH	DIS	RTF	DIR	NRIR	#	FACC	
11	OUT	JMP	FETCH	DIS	#	#	NRIR	ACCD	NFD	
12	GOTO	JMP	START	READ	RTF	MDOR	NRIR	#	FPC	
13	IF	JIF0	GOTO	DIS	#	#	NRIR	#	NFD	
14	FETCH	JMP	START	DIS	ONE	#	NRIR	PCD	FPC	INCR PC

Q: Could both addressing modes exist in this system?

Q: How many lines of microcode would be needed?

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 Am2901s.

DESIGN APPROACH:

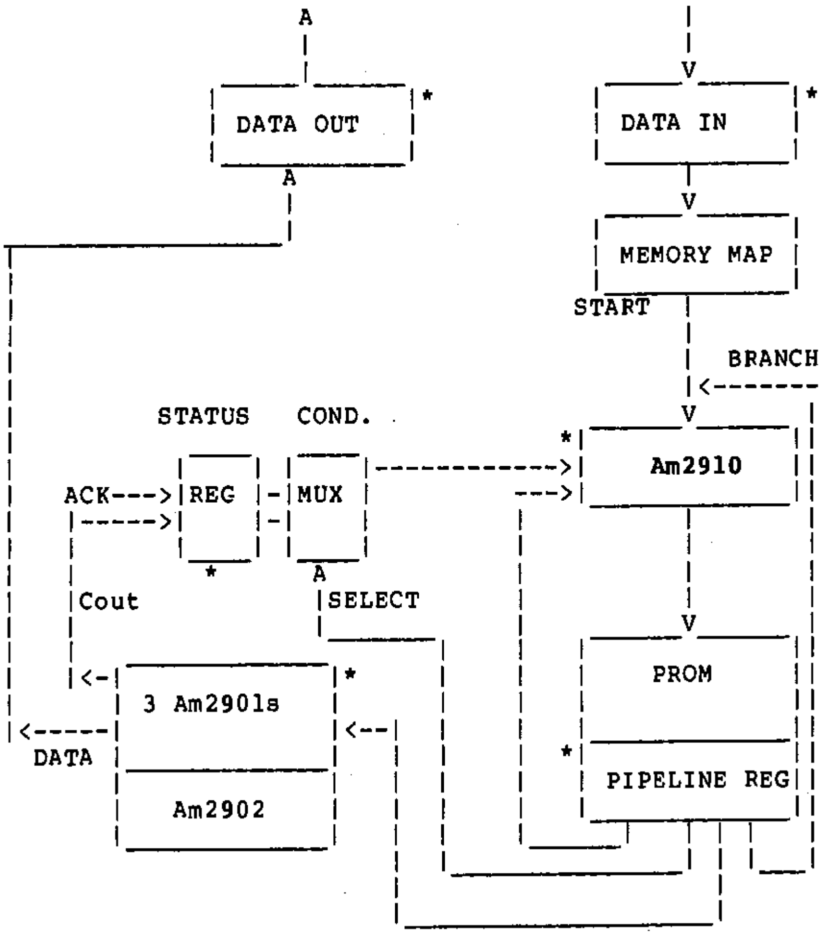
Use a pipelined PROM control memory, a status register (1-bit) and a memory map to decode the data input.

MICROPROGRAM DESCRIPTION:

```
START
    1. INITIALIZE REGISTERS
         $R_0$ <-- 0    $R_1$ <-- 0    $R_2$ <-- 0    $R_3$ <-- 0    $R_4$ <-- 0
NEXT:2. LOAD DATA-IN REGISTER
         DATA-IN <-- DATA-BUS
    3. IF DATA-IN = 0
          THEN $R_0$ <-- $R_0$ + 1
               IF $C_{n+4}$ = 1
                   THEN JUMP SUBO
                        $R_1$ --> DATA-OUT
         ELSE $R_1$ <-- $R_1$ + 1          ALL NUMBERS ARE POSITIVE
              IF $C_{n+4}$ = 1
                  THEN JUMP SUBO
                       $R_0$ --> DATA-OUT
    4. CASE BRANCH
       IF 0 < DATA-IN $\leq$ 5
           THEN $R_2$ <-- $R_2$ + 1
       IF 5 < DATA-IN $\leq$ 10
           THEN $R_3$ <-- $R_3$ + 1
       IF 10 < DATA-IN
           THEN $R_4$ <-- $R_4$ + 1
    5. BRANCH TO NEXT
```

```
SUB0:6. $R_2$ --> DATA-OUT
    7. IF $\overline{ACK}$
         THEN WAIT
    8. $R_3$ --> DATA-OUT
    9. IF $\overline{ACK}$
         THEN WAIT
    10. $R_4$ --> DATA-OUT
    11. IF $\overline{ACK}$
         THEN WAIT
    12. RESET REGISTERS
         $R_2$ <-- 0    $R_3$ <-- 0    $R_4$ <-- 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	0
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
* S U B R O U T I N E												
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

THE SOLUTION FOR A SINGLE MEMORY MAP WITH A TEST OF THE
Z STATUS BIT IS VERY SIMILIAR (ONE MICROWORD LONGER)

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	NA
1	CONT	#	#	AB	EXOR	RAMF	#	R1	R1	NOLOAD	NOLOAD	NA
2	CONT	#	#	AB	EXOR	RAMF	#	R2	R2	NOLOAD	NOLOAD	NA
3	CONT	#	#	AB	EXOR	RAMF	#	R3	R3	NOLOAD	NOLOAD	NA
4	CONT	#	#	AB	EXOR	RAMF	#	R4	R4	NOLOAD	NOLOAD	NA
NEXT	CONT	#	#	#	#	NOP	#	#	#	LOAD	NOLOAD	NA
6	CONT	#	#	DZ	OR	RAMF	#	#	#	ENABLE	NOLOAD	NA
7	CJP	NOTZ	DNOTZ	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
DISZ	CONT	#	#	AZ	ADD	RAMF	H	R0	R0	NOLOAD	NOLOAD	NA
9	JSUB	Cn+4	SUB0	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
A	CONT	#	#	AB	OR	RAMA	#	R1	R1	NOLOAD	LOAD	NA
WAITA	CJP	NOACK	WAITA	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
C	CJP	PASS	CASE	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
DNOTZ	CONT	#	#	AZ	ADD	RAMF	H	R1	R1	NOLOAD	NOLOAD	NA
E	JSUB	Cn+4	SUB0	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
F	CONT	#	#	AB	OR	RAMA	#	R0	R0	NOLOAD	LOAD	NA
WAITB	CJP	NOACK	WAITB	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
CASE	JMAP	#	#	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
L5	CJP	PASS	NEXT	AZ	ADD	RAMF	H	R2	R2	NOLOAD	NOLOAD	NA
GR5L10	CJP	PASS	NEXT	AZ	ADD	RAMF	H	R3	R3	NOLOAD	NOLOAD	NA
GR10	CJP	PASS	NEXT	AZ	ADD	RAMF	H	R4	R4	NOLOAD	NOLOAD	NA
* S U B R O U T I N E												
SUB0	CONT	#	#	AB	OR	RAMA	#	R2	R2	NOLOAD	LOAD	NA
WAIT1	CJP	NOACK	WAIT1	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
17	CONT	#	#	AB	OR	RAMA	#	R3	R3	NOLOAD	LOAD	NA
WAIT2	CJP	NOACK	WAIT2	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
19	CONT	#	#	AB	OR	RAMA	#	R4	R4	NOLOAD	LOAD	NA
WAIT3	CJP	NOACK	WAIT3	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA
1B	CONT	#	#	AB	EXOR	RAMF	#	R2	R2	NOLOAD	NOLOAD	NA
1C	CONT	#	#	AB	EXOR	RAMF	#	R3	R3	NOLOAD	NOLOAD	NA
1D	CONT	#	#	AB	EXOR	RAMF	#	R4	R4	NOLOAD	NOLOAD	NA
1E	CRTN	PASS	#	#	#	NOP	#	#	#	NOLOAD	NOLOAD	NA

TOTAL NUMBER OF MICROWORDS = 1E (HEX) = 30

TOTAL NUMBER OF BRANCH ADDRESS LINES = 5

FOR BOTH OF THE MICROPROGRAMS, IF THE SYSTEM IS TO BE RESET OR INITIALIZED BY PULL-UP RESISTORS ON THE PROM ADDRESS LINES [THE 2910 IS OUTPUT-DISABLED; THE PULL-UPS SHOULD BE 10K EACH LINE], THEN ADD ONE MICROWORD FOR THE "JZ" INSTRUCTION (JZ=0000).

PULL-DOWNS COULD BE USED AT THE INSTRUCTION LINE INPUT TO THE AM2910 (SEE HOTBACS CHAPTER 9; SUPER-16 COMPUTER).

A RESET ON THE PROM PIPELINE REGISTER TO START THE REGISTER WITH ALL ZEROS COULD BE USED.

AS AN EXERCISE, COMPLETE THE HARDWARE CONFIGURATION FOR THE VERSION OF THE MICROPROGRAM WHICH YOU CHOSE TO EXAMINE AND CONFIRM THAT THE MICROPROGRAM CANNOT BE FURTHER REDUCED.

SUMMARY OF THE TWO SOLUTIONS:

Using two maps requires a larger mapping PROM.

```
ADDRESS  4      |______________________|     5
---------/------>|     MEMORY MAP       |----/----> TO Am2910
---------------->|     32x5             |
PIPELINE SELECT |______________________|
```

It also requires one more bit in microword width.

Using the Z status as a test requires a larger status register and a larger conditional test multiplexer.

```
                STATUS REGISTER   CONDITIONAL MULTIPLEXER
FROM Am2901 MSS|________|     |________|
       -------->|F=0     |--->|Z       |
       -------->|Cn+4    |--->|Cn+4    |-----------> C̄C̄
  ------------>o|NOACK   |--->|NOACK   |
  EXTERNAL      |________|    |________|
  SIGNAL                           ^
                                   | MUX SELECT
                                   / 2
                                   |
```

It also requires one more bit in the microword to select the larger multiplexer.

EXERCISE

Redesign the data monitor, replacing the Am2901s with **Am29203s**. Everything else remains unchanged.

TEXT BOOK EXERCISES

The following exercises are coordinated with the Chapters in Bit Slice Design: Controllers and ALUs (White; Garland Press; NY:1980)

They are also sequenced on ED2900A, "Introduction to the 2900 Family", one of the seminars offered by the Customer Education Center of Advanced Micro Devices, Inc., Sunnyvale, CA.

EXERCISES - CHAPTER ONE

1. Obtain a current Am2900 Family Data Book and obtain a copy of the HOTBAC series. (This is the 9-chapter "How to Build a Computer" application note series produced by the Bipolar Applications Department of Advanced Micro Devices, Inc.)

2. List five disadvantages of using bit-slice versus an MOS FIS microprocessor.

3. List five advantages of using bit-slice versus an MOS FIS microprocessor.

EXERCISES - CHAPTER TWO

1. Using some typical devices, sketch the schematic for the improved-architecture CCU as developed in this chapter.

2. Make up the parts list for the CCU.

3. Compute the worst-case timing path for your design. This is your minimum clock-cycle time.

EXERCISES - CHAPTER THREE

1. For the architecture and the microword defined in section 3.3, write and diagram sample code as was done to demonstrate unconditional and conditional branching. Demonstrate subroutine call and return.

2. Using the Am29811 (latest data sheet), a condition code multiplexer (MUX), and the Am2909/11 (also the latest data sheet), write microcode to call a subroutine if the test input is true, and call a second subroutine if it is false. The first subroutine contains a five (5) step loop which executes sixteen (16) times. The second subroutine contains a five (5) step loop which repeats until a second test is true. Show how to return to the main program. NOTE: Only the sequence control portion of the microcode is of interest.

3. Consider using the Am29803 in place of the condition code MUX. Is this reasonable? And, if not, why not?

EXERCISES - CHAPTER FOUR

1. Repeat exercise 3.2 using the Am2910 in place of the Am29811 and Am2911.

2. What are the changes if a microroutine is called on the occurrence of an interrupt (condition tested = true), rather than a subroutine in the above exercise?

3. Sketch the CCU which uses an Am2914 and a vector mapping PROM in addition to the Am2910, which can support firmware-level interrupts and handle interrupts via subroutines (CJS, etc.) <u>or</u> microroutines (CJV, etc.).

4. Refer to the latest Am2910 data sheet.

   a. What is the worst-case delay for data to pass from the D_i inputs to the Y_i outputs?

   b. From the $\overline{CC}$ input to the Y_i outputs?

   c. From the I_i inputs to the output-enable controls?

   d. What is the worst case set up and hold time required for the internal register?

   e. For the microPC?

   f. For the instruction bits?

   g. If the previous instruction alters the register/counter, what happens in terms of time delay?

EXERISES - CHAPTER FIVE

1. Draw the schematic for a 16-bit simple computer as defined in section 5.5, using the Am2901 and Am2910 devices.

2. Develop all of the microcode required to support the instruction set given in section 5.5 (make simplifying assumptions concerning memory control, external devices, for the time being).

3. Enhance the schmatic of exercise 5.1 so that the shift and rotate instructions of section 5.7.6 are supported. Do not use register instructions yet.

4. Supplement the microcode of exercise 5.2 to include shift and rotate instructions. Design those instructions.

5. Add the necessary hardware to allow register arithmetic to be possible with your computer. Assume both direct operand addressing with a default register used as the accumulator and general register addressing are supported. Assume also that the register addresses may come from the CCU pipeline or from the instruction register (use an AB MUX).

6. Write the microcode to support the hardware additions made in exercise 5.5.

EXERCISES - CHAPTER SIX

1. Repeat exercise 5.3 using the Am2903 in place of the Am2901.

2. Repeat exercise 5.4 using the Am2903 in place of the Am2901.

3. Repeat exercise 5.5 using the Am2903 in place of the Am2901.

4. Repeat exercise 5.6 using the Am2903 in place of the Am2901.

EXERCISES - CHAPTER SEVEN

1. Replace the status register you should have had in your computer, as well as the four shift-rotate multiplexers, with an Am2904.

2. Alter your microcode for this change.

EXAM ANSWERS

EXERCISE SOLUTIONS

2900 FAMILY STUDY GUIDE
EXERCISE SOLUTIONS
SELECTED ANSWERS FOR THE EXAM

1.

LABEL/ ADDR	29811 INSTR	MUX SEL	BR ADDR	29803 INSTR
i	CJP	A	i + 2	NO TEST
i + 1	JP	#	350*	T_3T_1
i + 2	JP	#	360*	T_2T_0
j	CJP	B	j + 2	NO TEST
j + 1	JP	#	370	$T_2T_1T_0$
j + 2	JP	#	380	$T_3T_2T_1$
350	JP	#	20	NO TEST
351	JP	#	200	NO TEST
352	JP	#	10	NO TEST
353	JP	#	20	NO TEST
360	JP	#	10	NO TEST
361	JP	#	200	NO TEST
362	JP	#	30	NO TEST
363	JP	#	40	NO TEST
370	JP	#	100	NO TEST
371	JP	#	200	NO TEST

* address must have 0 as final HEX digit if LSS 2909 attached to 29803

ETC.

	2910 instr.	mux sel.	br addr.	
2.	LDCT	#	5	LOAD COUNTER
	PUSH	A	9	RELOAD BASED ON A
	S1:CONT	#	#	
	S2:CONT	#	#	
	S3:RFCT	#	#	
OR				
	LDCT	#	9	
	PUSH	$\overline{A}$	5	
	S1:CONT	#	#	
	S2:CONT	#	#	
	S3:RFCT	#	#	
3.	PUSH	PASS	N	
4.	CONT	#	#	
5.	TWB	TEST	72	

6. address 64

7. The pipeline register at the output of the PROM allows the FETCH/EXECUTE cycles to overlap and the two longest time paths are as balanced as possible (even length). A register placed at the output of the Am2910 and input to the address lines of the control memory, instead of at the output of the control memory, would allow a shorter microcycle than if no pipeline registers were used but the two time paths thus created are not balanced, resulting in a longer 'critial path'.

8. 80 - 85ns

9. from the Am2910 itself;
   from special bits in the microword
   by decoding the instruction lines with SSI

10. 2910: PASS line on MUX input (grounded)
          $\overline{CCEN}$ = HIGH
    29811: PASS line on MUX input (tied high)

11. 16 x 4

12. no

13. 16 x 4

14. yes, use the Am29705 to expand the Am2903

15. 16 x 4

16. yes, use the Am29707 to expand the Am2903

17. The A and B RAM ports drive separate 4-bit latches which hold the RAM data while the clock input is low. They pass data while the clock is high.

18. yes

19. on the rising edge of the clock

20. 50pf

21. no - the DA_{0-3} pins are input only on the Am2903 and bidirectional on the Am29203. Z and $\overline{I}_{EN}$ also are slightly different in order to support byte operations better.

22. Am29203

23. Am29203

ANSWERS TO TRUE OR FALSE QUESTIONS

1. TRUE - ie. HEX-29's design

2. TRUE

3. FALSE - look at RPCT, RFCT, TWB

4. TRUE

5. TRUE - designed that way. Am2901 can do it too.

6. FALSE - however, it may have a higher throughput

7. TRUE

8. FALSE - it is possible, 50:1 parts ratio with a 2:1 speed improvement, but practical??

9. TRUE - add a few items to the SUPER-16 and you have it

10. FALSE - it does support real time interrupts, but at a high cost in parts, power, real estate and conceptual simplicity

11. TRUE - connect Vector Map to Data Bus

12. TRUE - connect Vector Map to D_i of the Am2910

13. FALSE

14. TRUE - an LS family, it handles 1 TTL load

15. TRUE - but shaky; requires an extra pipeline register over the typical CCU design AMD usually presents. New parts will be faster and do it easier.

16. FALSE - some disk controllers are computers in their own right

17. TRUE - for subroutine calls drive the output enable lines from the microinstruction rather than the Am2910 pins

	2910	MUX SEL	ADDR	2914 INSTR	$\overline{\overline{OE}}_{VECT}$
subroutine:	CJS	ANY TEST	#	READ VECT	EN
nonsubroutine:	CJV	ANY TEST	#	READ VECT	(from 2910)

18. FALSE - the 2909 and the 2911 do

19. FALSE - the 29803 attaches to the 2909

20. TRUE

21. FALSE - 2909, no; 2911 and 2910, yes

1.

2. $\overline{OE}$ = LOW

3. shared CCUs
on executing JZ with pull-ups on address lines (10K)

4. to drive an external counter for the DO X TIMES loop

5.

6. active high

7.

ADDRESS	29811 INSTR	MUX SEL	BR ADDR	
10	CONT	#	#	
11	CONT	#	#	
12	CJP	TEST	40	NOTE: a symbolic label should be used!
13	CONT	#	#	
14	PUSH	PASS	9	
15	CONT	#	#	
16	RFCT	#	#	
17	CONT	#	#	
40	CONT	#	#	
41	CJS	PASS	60	unconditional jump to subr.
42	CONT	#	#	
60	CONT	#	#	
61	CONT	#	#	
62	CJP	TEST	80	
63	CRTN	PASS	#	
80	JP	#	63	

1. I_i = 0, all i; or $\overline{OE}_i$ = HIGH

2. select which of several tests is to be done, the output enables tri-state disable the 29803

3. 2 ↑ 2 = 4; if a 29811 is used, then 5: TEST, and the T_i

4. C, 9, 6, F

5.

ADDRESS	29811 INSTR	MUX SEL	BR ADDR	29803 INSTR	
11	CONT	#	#	NO TEST	
12	JP	#	##0	$T_3T_2T_1$	
##0	JP	#	40	NO TEST	Branch Table
##1	JP	#	60	NO TEST	
##2	JP	#	140	NO TEST	
##3	JP	#	10	NO TEST	
##4	JP	#	33	NO TEST	
##5	JP	#	45	NO TEST	
##6	JP	#	45	NO TEST	
##7	JP	#	45	NO TEST	

NOTE: address ##0 should be a symbolic label whose value after assembly (AMC SYS/29) should be one with zero as the last HEX digit. The 29803 should be tied to the least significant 2909 (LSS). All of the branch table addresses should also be given as symbolic labels. Numerics were shown here for demonstration only.

1. Worst case shows address = LOW to be a state requiring examination. At 0.5V the Am2911 provides 12mA (Y_i only) and the 32 loads require -8mA.

2. no

3. 5pf/load = 5x32 = 160pf
Am2909/2911/2910 rated for 50pf load

4. (160 - 50)pf * 0.1ns/pf = 11ns degredation

5. nothing, adding a buffer adds <u>its</u> delay.

6. 4Kx32 bits from 0.5Kx8 = 8x4 array
still 32 loads as above

7. 4kx64 bits = 8x8 array
64 loads is not possible without buffer/drivers on address lines (limit is 12mA on Y_i at V_{LOW}).

1. $\overline{RLD}$ = LOW causes the data on the D_i lines of the 2910 to be loaded into the register/counter.

2. C_i = HIGH allows the incrementer to increment normally.

3. $\overline{CCEN}$ = HIGH makes $\overline{CC}$ a don't care and changes all conditional instructions into unconditional instructions.

4. $\overline{FULL}$ should be tested during program development and by any debug/diagnostic routines.

5. Yes - but only one may be active on a bus at any one time.

6. realtime interrupts

7. a) start up using pull-ups on the Y_i lines (JZ placed at address FFF)
   b) switching between microprogram address sources

8. 5 deep

9. no

10. 2909/2911s have 4 deep wrap-around stack

11. 8 <--- top overridden
    4
    3
    2
    1

12. same as 29811 exercise #7

13.

2910	29811
$\overline{CCEN}$ alters conditional instr.	no $\overline{CCEN}$
TWB	no TWB, JP instead
$\overline{CC}$ is active low	TEST is active HIGH
$\overline{FULL}$ pin for stack	no full pin
$\overline{OE}_{VECT}$ output	must decode others

14. CJP (2910) branches on $\overline{CC}$ = LOW

15. Yes, but it is NOT recommended. Requires some means of selecting the source of the branch address (pipeline register or 29803)

16. TRUE

17. RFCT frees the branch address field for other uses
    RPCT leaves the stack alone

18. TWB; contains a "dead man timer" to limit loop repetition

	LABEL/ ADDR	2910 INSTR	COND MUX SEL	$\overline{CCEN}$	BR ADDR/ COUNTER
19.	i	CRTN	TESTA	*	#
20.	i	CRTN	PASS	#	#
21.	i	CRTN	#	H	#
22.	i	LDCT	#	#	9
	i + 1	RPCT	#	#	i + 1
23.	i	PUSH	PASS	#	9
	i + 1	RFCT	#	#	#
24.	i	PUSH	#	H	9
	i + 1	RFCT	#	#	#
25.	i	PUSH	FAIL	*	#
	i + 1	LOOP	TESTB	*	#
26.a.	i	PUSH	PASS	#	19
	i + 1	TWB	TESTC	*	FAILADR
26.b.	j	PUSH	#	H	19
	j + 1	TWB	TESTC	L	FAILADR

note 1.

means DON'T CARE

* means signal either not present in pipeline (tied LOW) or signal must be driven LOW

	LABEL/ ADDR	2910 INSTR	COND MUX SEL	$\overline{CCEN}$	BR ADDR/ COUNTER	$\overline{RLD}$ (added)
27.	i	LDCT	#	#	ADR1	
	i + 1	JRP	TESTD	*	ADR2	
28.	i	LDCT	#	#	SUBA	
	i + 1	JSRP	TESTD	*	SUBB	
29.	i	CJPP	TESTE	*	ADR3	
30.	i	. . . note 2			SUBA	L
	i + 1	. . .				H
	i + 2	. . .				H
	i + 3	JSRP	TESTD	*	SUBB	#
31.	13	CJS	TESTC	*	40 (use label)	
32.	13	CJS	X	H	40	
33.	13	CJS	TESTC	*	40	L
34.	13	CJS	TESTC	*	40	note 1.
35.	13	CJS	TESTC	L	40	L

(The above assumes that the test was desired and that no register load is to occur. The microinstruction will vary with the application.)

note 1. C_{in} may NOT be controlled from the microinstruction!

note 2. no RFCT, LDCT, RPCT, TWB, or PUSH(PASS) may occur in this code section

FLOWS:

NORMAL CJS EXECUTION:

```
        |
        |
     11 *
        |               STACK
        |    PASS     --------    PUSH IF TESTC IS TRUE
     12 *  ---------->|  14  |
        | /           --------
        |/    PASS
 CJS 13 *---------->* 40
        |           |
        | FAIL      |
     14 *  |        * 41
        |  |        |
        |  V        |
     15 *           * 42
        |           |
        |           |
        V           V
```

CJS EXECUTION WITH $\overline{\text{CCEN}}$ = HIGH

```
        |
        |
     11 *
        |               STACK
        |    DO IT    --------    PUSH
     12 *  ---------->|  14   |
        | /           --------
        |/   GO TO
 CJS 13 *---------->* 40
                    |
                    |
     14 *           * 41
        |           |
        |           |
     15 *           * 42
        |           |
        |           |
        V           V
```

CJS EXECUTION WITH $\overline{\text{RLD}}$ = LOW

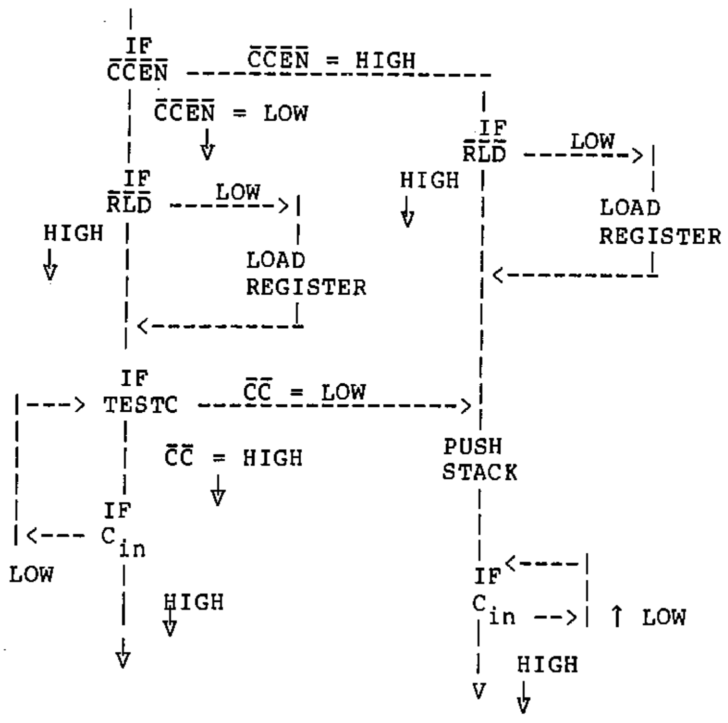
```
                |
                |
             11 *
                |               STACK
                |    PASS     --------    PUSH IF TESTC IS
             12 *  ---------->|  14  |    TRUE
                | /           --------
                |/    PASS
         CJS 13 *---------->* 40
              / |           |
         ANY /  | FAIL      |
            /14 *  |        * 41
 REGISTER  /    |  |        |
 ---------/     |  V        |
 |  40   |   15 *           * 42
 ---------      |           |
                |           |
                V           V
```

CJS EXECUTION WITH Cin SWITCHED:

```
        |
        |
     11 *
        |               STACK
        |    PASS       --------   PUSH IF TESTC IS TRUE
     12 *  ---------->|  13  |
        | /           --------   NOTE THE VALUE!
        |/    PASS
 CJS 13 *---------->* 40
C  =LOW |          |     CONTINUE AFTER C   GOES HIGH
 in                                      in
        | FAIL     |
     14 *  |       * 41
        |  |       |
        |  V       |
     15 *          * 42
        |          |
        |          |
        V          V
```

YOU SEE HOW THIS CAN LEAD TO PROBLEMS! TWB FOR EXTERNAL SIGNAL TESTING IS A MUCH BETTER CHOICE.

FLOWCHART OF CJS DECISION PROCESS IN THE PRESENCE OF $\overline{CCEN}$, $\overline{RLD}$, C_{in}:



1. 10 address lines

2. $\overline{CS}$ $\overline{WE}$

3. chip is 1Kx4; therefore, 4 rows, 8 cols required for a 4Kx32 bit memory.

   use decoder to drive chip selects

4. 32 loads: 10μA input load current
   5pf input capacitance | per load

   This puts an 11ns speed degredation on the Am2910 timing.

5. 64 loads, no

6. must be buffered with drivers on the address lines

1. 8

2. active low

3. 1

4. yes

5. Read Vector

6. used to clear last vector read

7. when a Master Clear, Clear All Interrupts, or Clear Interrupt Last Vector Read is executed

8. the binary value indexes the highest priority nonmasked interrupt request currently present

9. no, one greater

10. 5, 6, 7

11. 1

12. No, PASS ALL is an output internally generated by the other signals (you tell that by the absence of a 'o' on the begining of the line).

1. no
2. no
3. F, the ALU output
4. no
5. 2's Cmpl., can be hooked up for 1's Cmpl with extra SSI
6. no, 2 microcycles
7. Data Book, edition Jan 1980, pg 2-19 (MPR-020)

2901

ADDR	2910	MUX	BR ADDR	A	L	U	RA	RB	Cin	ROTATE
	INST	SEL	COUNTER	SRCE	FUNC	DEST	ADDR	ADDR	SEL	MUX SEL
8.	#	#	#	ZA	ADD OR	RAMD	R6	R6	L #	RAM0TORAM3
9.	#	#	#	AB	ADD	RAMU	R6	R6	Cn	RAM3TORAM0
10. yes										
	#	#	#	ZA	ADD	RAM	#	Ri	H	#
11.i	CONT	#	#	DZ	OR	QREG	#	#	#	#
i+1	#	#	#	DQ	ADD	QREG	#	#	L	#
12.i	CONT	#	#	AB	ADD	QREG	RA	RB	L	#
i+1	#	#	#	ZQ	OR	RAM	#	RC	#	#
13.i	LDCT	#	2	AB	ADD	RAMU	R6	R6	Cn	RAM3TORAM0
i+1	RPCT	#	i+1	AB	ADD	RAMU	R6	R6	Cn	RAM3TORAM0
14.	CONT	#	#	ZB	ADD	RAMA	R15	R15	H	#

Which instruction lines provide or do the following?

Am2901	$I_8I_7I_6$	$I_5I_4I_3$	$I_2I_1I_0$
Select of MUX at R(ALU)			X
Function select of ALU		X	
Enables the Q register	X		
Select of Output MUX	X		
Inhibit at R or S			X
Select of MUX at RAM input	X		
Select of MUX at QREG input	X		
RAM enable	X		
RAM_0 enable	X		
RAM_3 enable	X		
Q_0 enable	X		
Q_3 enable	X		
Select of MUX at S(ALU)			X

1. the sign bit is <u>NOT</u> affected during an arithmetic up/down shift
2. yes
3. no
4. yes
5. 8 at present; more on the 29203; plus space for expansion in the future
6. yes
7. yes
8. yes
9. yes
10. yes
11. I_0-I_4 all LOW means special functions; the, I_5-I_8 select which function is to be done
12. no
13. no
14. yes; SLN uses different meanings for C_{n+4} and OVR and you <u>must</u> execute SLN with $\bar{I}_{EN}$ disabled before beginning the normalize algorithm.

LABEL ADDR	2910 INST	MUX SEL	BR ADDR COUNTR	A SRCE	L FUNC	U DEST	RA ADDR	RB ADDR	Cin SEL	ROTATE MUX SEL	RC ADDR
15. i	#	#	#	DADB	R+S	F->Y	#	#	L	#	
16. i	#	#	#	RAMAB	R+S	F->RAM	RA	RB	L	#	RC
17. i	LDCT	#	2	RAMAB	R+S	2F->RAM	RA	RB	Cn	RAM3TORAM0	#
i+1	RPCT	#	i+1	RAMAB	R+S	2F->RAM	RA	RB	Cn	RAM3TORAM0	#
18. i	#	#	#	RAMAB	R+0	F->RAM	R15	R15	H	#	
19. i	#	#	#	DA#	R+0	F->Q	#	#	L	#	
20. i	#	#	#	RAMAB	R+0	2F->RAM	R2	R2	H	RAM3TORAM0	#
21. i	LDCT	#	15	RAMA#	R+0	F->Q	R2	#	L	#	
i+1	RPCT	#	i+1	LOW	LOW	SPF0	R1	R0	L	#	
22. i	LDCT	#	14	RAMA	R+0	F->Q	R2	#	L	#	
i+1	RPCT	#	i+1	LOW	LOW	SPF2	R1	R0	L	#	
i+2	#	#	#	LOW	LOW	SPF6	R1	R0	Z	#	
23. i	#	#	#	RAMB	S+0	FQ/2-> RAMQ	#	R2	#	RAM0TOQ3	
24. i	CONT	#	#	RAMAB	R+S	2F->RAM	R2	R2	L	RAM3TORAM0	#
i+1	#	#	#	RAMB	S+0	F->Q	#	R2	L	#	

R+0 IS FASTER THAN S+0 AND OR IS FASTER THAN ADD FOR PASSING DATA
= DON'T CARE
16. is the only microinstruction using three addresses

Which group of instruction lines or enables controls the following?

Am2903	$\bar{E}_A I_0 \overline{OE}_B$	$I_4 I_3 I_2 I_1$	$I_8 I_7 I_6 I_5$
Input MUX at R	X		
RAM enable			X
Switch to special functions		X	
ALU function		X	
WRITE enable			X
Q_0 enable			X
Q_3 enable			X
SIO_0 enable			X
SIO_3 enable			X
Q Register enable			X
Input MUX at S	X		
Chip function (during special functions)			X
DB_{0-3} enable	X		
Status output select			X
Inhibit at R and S	X		
Q input MUX			X
RAM input MUX			X

a. $\bar{G} \oplus PC_n$; $\emptyset$; $Q_3 \oplus Q_2$; $F_3 \oplus F_2$

b. 5

c. $\bar{P}$; $\emptyset$; $C_{n+3} \oplus C_{n+4}$; $Q_2 \oplus Q_1$; $F_2 \oplus F_1$

d. 4

e. F_3; $F_3 \oplus S_3$; Q_3; $\bar{G}$

f. 5

g. $Y_i = LOW|_{all\ i}$ Q_0 $Q_i = LOW|_{all\ i}$ Sign compare F/F
$Y_i \hat{} Q_i = LOW|_{all\ i}$

h. FALSE - not on the Am29203

i. Yes

j. Sets status register with values relating to functions

k. $\overline{WRITE}/MSS|_{LSS}$ ----> $\overline{WE}$ on all chips

1. 8

2. 5

3. one connects to a decoder which is connected to the C address line C_4 and enabled by the $\overline{\text{WRITE}}/\overline{\text{MSS}}$ of the LSS 2903; the other connects to the clock line, C_p.

4. $\overline{\text{LE}}$ is connected to C_p also.

5. yes (watch your timing)

6. yes, just as 2903 registers are loaded, the only difference is the address line.

7. a. 6 each for A, B, C addresses
   b. 4 each for A, B, C addresses plus some PSW bit to control chip select

	Am2904 $I_9I_8I_7I_6$	$\overline{SE}$	Am2901 $I_8I_7I_6$
1.	0	0	5
2.	6	0	4
3.	7	0	6
4.	A	0	7

			Am2903 $I_8I_7I_6I_5$
5.	2	0	5
6.	5	0	3
7.	E	0	1
8.	E	0	0

9.

I_{12}	I_{11}	I_5	I_3	I_2	I_1	
1	1	0	0	#	#	
1	1	0	#	#	1	any of these
1	1	0	#	1	#	

2904

	$\overline{OE}_Y$	$I_5I_4I_3I_2I_1I_0$	$\overline{CE}_\mu$	$\overline{CE}_M$	$\overline{E}_Z\overline{E}_C\overline{E}_N\overline{E}_{OVR}$	$\overline{OE}_{CT}$
10.	X	00	0	0	0	H
11.	H	02	0	0	0	H
12.	H	01	0	1	X	H
13.	H	03	0	1	X	H
14.	H	01	1	0	B	H
15.	H	03	1	0	B	H
16.	H	04	1	0	A	H
17.	X	00 10	1	1	X	L

18. Perform the test but also perform MSR --> μSR; Y_i --> MSR.

19.	X	36	1	1	X	L

20. Perform the test but also perform I_i --> μSR.

21.	X	2A	1	1	X	L

LABL	2910	M U	BR	A	L	U	A	B	2904				
ADDR	INST	X	CNTR	SRCE	FUNC	DEST	ADDR	ADDR	C_{in}	µSR	MSR	TEST	MUXSEL
22.i	CONT	X	X	ZB	ADD	RAMA	R15	R15	H	X	X	DIS	X
23.i	LDCT	X	2	AB	ADD	RAMU	R6	R6	I_C	X	X	DIS	RAM3TORAM0
i+1	RPCT	X	i+1	AB	ADD	RAMU	R6	R6	I_C	X	X	DIS	RAM3TORAM0
24.i	X	X	X	DZ	ADD	RAMA	R6	R6	L	X	X	DIS	X

25. C_{in} is replaced by $I_{12} I_{11} + I_5 I_4 I_3 I_2 I_1 I_0$ of the Am2904

Status registers added as needed $I_5I_4I_3I_2I_1I_0 + \bar{E}_i + \bar{C}\bar{E}_i$s

Multiplexer selection for shift control done on one part rather than several separate multiplexers and then attempting to encode their controls.

Conditional test field added (may overlap MUX SEL field or may even replace it, depending upon the design).

1. True

2. False

3. True - byte or word operation all almost every instruction

4. True - using either the instruction lines, I_i, or the T_i lines

5. True

6. False - use an Am2904 for emulations (bit settable status registers)

7. True - requires two microwords (two microcycles if 1 microinstruction executes in 1 microcycle)

8. True - I_i, ACC, D Latch, or RAM

9. True - control timing and use an external MUX similiar to the procedure used by the Am2903/Am29203

10. True

11. False! - remember the branch on previous - NOP two step required by the double pipelined Am2910 - Am2901/2903/29203 "typical" CPU (ED2900B)

12. True

13. True

14. True with reservations - the D latch can be used but requires some tricky timing, you could generate a race condition if the D latch is also a source for the operation. The D latch is <u>not intended</u> to be a destination, and its use as such is <u>not recommended</u>. Normal destinations: RAM, ACC or NONE.

15. True

16. False - D_{SE} (D sign extend with bit 7 extended) is used for Two's complement arithmetic

17. True

18. True

19. True

20. False - also Q_C, $C_N \oplus Q_{OVR}$ (Q_i is used to indicate the status register contents in the Am29116 literature; not to be confused with the Q register of the RALUs)

21. True

22. True

23. True - also goes to Y_i

24. True - watch timing

25. False - ACC only, instruction has common source/ destination field

26. True - none and 0 - 15
27. True
28. True
29. True - AMD survey
30. True
31. False!
32. True
33. True
34. True
35. False - if you need this use an Am2904
36. False - this is the one source which disallows loading the ACC
37. True
38. True

ANSWERS TO FILL IN OR ANSWER SECTION:

39. Z, C, N, OVR, LINK, FLAG1, FLAG2, FLAG3

40. There are 12 condition code test signals:
    the 8 status bits themselves (one at a time)
    plus: $N \oplus OVR$ $[N \oplus OVR] + Z$
    $Z + \bar{C}$ LOW

41. Yes - by using the T_i lines as input. This requires a wider microword so that both the T_i instruction lines and the I_i instruction lines are present.

42. lower <u>only</u>

43. Yes - <u>very carefully</u>! Requires a register between the microcontroller and the control storage (usually a PROM memory) as well as the pipeline register at the control storage output. 125ns is more easily achieved. Timing studies will be more detailed when the part is released (end of 1980).

44. RAM, ACC, D, I, ZERO

45. NO

46. RAM-ACC, RAM-I, D-RAM, D-ACC, ACC-I, D-I

47. Rotate is UP only:

    1100 1010 0011 0101 n=2 becomes 0010 1000 1101 01<u>11</u>

48. Yes

49. Yes - watch your timing.

50. 0000

51. 0001

52. Bit 15 does not participate in the byte mode.

53. The program space is larger for the faster Am29116.

54. Yes - but it was not intended for this application.

55. Yes

56. Yes - the barrel shifter works in the byte or word mode.

57. Yes - 16-bit ALU

58. Yes - but it was not intended for this application.

59. If the mask bit i is zero in a ROTATE-MERGE instruction, the ith bit of the U operand is passed to the destination.

60. If

```
   U = 0011 0001 0101 0110
   R = 1010 1010 1010 1010
MASK = 0101 1010 0110 1001
```

and n = 4
the bit pattern that is produced by a word mode ROTATE-MERGE instruction is 000 111 000 1010.

61. If the highest bit position with a one (1) is bit position 7, and the mask is 1010_{HEX}, 0009_{HEX} is produced by a word prioritize instruction.

62. Repeating for byte mode prioritize, same answer as 61.

63. For LOAD $\bar{2}^N$, ZERO --> bit N, ONE --> all other bits in destination.

64. To do a word rotate down five bit positions, do a rotate up of n = 11 [16 - 5 = 11].

```
Example:   0000 1111 0101 1010
           1101 0000 0111 1010
```

65. QLINK is the linkage bit for shift operations; also for CRC instructions (serial input).

66. You CANNOT set/reset the ALU status bits one by one as with the Am2904 Machine status register. They function more like the microstatus register.

67. You CAN set/reset the LINK and FLAGi status bits one at a time.

68. In byte mode, the lower 4 bits of the status register are loaded.

69. The instructions which do **NOT** cause the ALU status bits to be updated are: NOP, status register instrcutions, save status, test status, or any instruction if either

$\overline{SRE}$ or $\overline{IEN}$

are not LOW.

70. The upper four status bits (LINK, FLAG1, FLAG2, FLAG3) are changed during status set/reset; status load (word mode only); plus QLINK is updated after each shift.

Chapter 1

1.

2. larger number of parts - hardware
   more difficult programming
   more levels of programming
   more external connections
   multiple levels of documentation (AMC SYS/29 helps)
   longer design time

3. faster
   can custom tailor to the application via microprogramming
   emulations are easier - more flexible connectibility
   interfaces directly to TTL
   allows wider (>16 modulo 4) data path

Chapter 3

1.

2. Sample code:

LABEL	ADDRESS	Am29811 INSTR	COND MUX	BRANCH ADDRESS
	:			
	a	CONT	#	SUB1
	a+1	JSRP	T1	SUB2
	a+2	:		
	:			
SUB1	b	PUSH	PASS	5
	b+1	CONT	#	#
	b+2	CONT	#	#
	b+3	CONT	#	#
	b+4	CONT	#	#
	b+5	RFCT	#	#
	b+6	CRTN	PASS	#
	:			
SUB2	c	PUSH	FAIL	#
	c+1	CONT	#	#
	c+2	CONT	#	#
	c+3	CONT	#	#
	c+4	CONT	#	#
	c+5	LOOP	T2	#
	c+6	CRTN	PASS	#
	:			

Chapter 4

1. Same as 3.2 with the Am2910 instruction field replacing the Am29811 field

2.
```
     |         |Am2910 | COND| BRANCH  |Am2914
LABEL| ADDRESS| INSTR |  MUX| ADDRESS | INSTR
----------------------------------------------
       :                                 |
       a       CONT     #      #         -----added field
     a+1       CJV      T1     #       READVECT  <---
     a+2        :
       :
VEC1   b       PUSH    PASS    5
     b+1       CONT     #      #
     b+2       CONT     #      #
     b+3       CONT     #      #
     b+4       CONT     #      #
     b+5       RFCT     #      #
     b+6       CJP     PASS   a+2       <------------
       :
VEC2   c       PUSH    FAIL    #
     c+1       CONT     #      #
     c+2       CONT     #      #
     c+3       CONT     #      #
     c+4       CONT     #      #
     c+5       LOOP     T2     #
     c+6       CJP     PASS   a+2       <--------------
```

4.a) 20ns
b) 43ns
c) 51cns
d) 24ns
e) 58ns
f) 104ns
g) the C_p to Y_i time is 25ns longer

☆☆☆☆☆ ***THE END*** ☆☆☆☆☆