## SECTION VI

## PROGRAMMING EXERCISES

### INTRODUCTION

The Am2900 Evaluation and Learning Kit is intended to teach the basics of microprogramming to the hardware design engineer. The exercises are geared to the objectives of demonstrating the uses of the Am2901 and Am2909 and to involve the user in the microprogramming of these devices. In addition, it is intended to allow evaluation of the Am2901 and Am2909 in an application environment. As such, the learning exercises are divided into two sections. These include static exercises and dynamic exercises. The static exercises are performed using the toggle switches and momentary switches available on the printed circuit board. The user clocks the system by hand observing the various states of the machine at each point. The dynamic tests are set up using the switches on the printed circuit board but are evaluated by using an external pulse generator and oscilloscope. This allows the Am2901 and Am2909 to be tested under operating conditions. In this fashion, the student can learn microprogramming techniques in the SINGLE STEP CLOCK mode and can evaluate the components in the PULSE GENERATOR mode, bearing in mind that the kit has not been designed as a small computer. However, once the user understands the Am2901, Am2909 and the principles of microprogramming, then the design of a computer becomes obvious.

The static test exercises are divided into three basic groups. The first group is intended to familiarize the engineer with the Am2901 capabilities. The second group is intended to familiarize the engineer with the Am2909 functions. The third group of exercises that can be generated appear almost infinite. The exercises presented here are only representative and the student is encouraged to write numerous additional exercises to evaluate any specific parameter or feature in which he is interested.

In the dynamic testing mode, several exercises are presented such that the student can evaluate typical switching characteristics. Many of the specified parameters of the Am2901, Am2907, Am2909 and Am2918 can indeed be measured using the Am2900 Kit. It is not possible to measure some of the parameters due to the complex nature of the testing required.

### USING THE EXERCISES

The microprogramming exercises are contained on the Am2900 Kit microprogramming worksheets following this discussion. These worksheets show the specific "logic one" and "logic zero" codes that must be entered into the microprogram memory in order to execute the various microinstructions so as to perform a function (HIGH = logic one). The results of the execution of these micro-

instructions is described in the accompanying text.

The Am2900 Kit programming worksheet is divided into two main parts. The top half of the worksheet shows a map of the entire 16-word by 32-bit microprogram memory. The logic ones and logic zeros associated with the fields of each microprogram memory word can be entered into the worksheet. Any blank space on the worksheet indicates a "don't care" condition. The bottom half of the worksheet shows a functional block diagram representation of the Am2900 Kit. This block diagram is intended to allow the user to sketch the path of a microinstruction so he may more easily understand the actual event being executed. The use of these two sections of the worksheet is demonstrated in the exercises.

## PROGRAMMING THE MICROPROGRAM MEMORY

The recommended procedure for using the exercises as described with the Am2900 Kit is to load all the required data into the microprogram memory. For example, all six words of Exercise 1 should be loaded into the microprogram memory before executing these instructions. The procedure to be used in loading the microprogram memory is as follows:

1. Connect the 5V power supply to the Am2900 Kit and apply power.
2. Set the RUN/LOAD SELECT switch (S12) to the LOAD position.
3. Set the MEMORY ADDRESS switches (S8-S11) to the decimal zero position.
4. Set the RAM & MUX SELECT switches (S1-S3) to the decimal zero position.
5. Set the MEMORY DATA switches (S4-S7) to the required bit position.
6. Depress the MEMORY LOAD switch (S14) to enter the data.
7. View the MICROWORD MEMORY LED's to be sure the correct word has been written into the microprogram memory.
8. Repeat steps 3-7 for each decimal position of the RAM & MUX SELECT switches until all eight fields (decimal 0-7) have been loaded.
9. Repeat steps 3-8 for each microprogram memory address word until the entire 16 words (word 0-15) of the microprogram memory have been loaded.

Needless to say, any blank entries ("don't care" conditions) may be omitted. When this procedure is followed, all possible 128 four-bit fields in the microprogram memory are easily loaded.



The above description has sequentially loaded the 8 four-bit fields of the microprogram word, one word at a time. The user may wish to load all 16 words of each field sequentially. That is, the RAM & MUX SELECT switches are set to decimal 0 and each MEMORY ADDRESS is loaded with the appropriate word for the entire 16 words of the memory. Advanced Micro Devices' Applications Engineers have found both techniques to be convenient depending on the particular exercise.

Once the microprogram memory has been loaded, the exercise should be executed as described in the accompanying text with the programming worksheet.

## CONNECTING THE PULSE GENERATOR

The Am2900 Evaluation and Learning Kit is designed such that it can be easily driven from a pulse generator. Two turret terminals are provided near resistor R9 so that a coaxial cable can be soldered to the board. One of the turret terminals is marked "GND" and connects directly to the ground bus of the system. A 51 ohm resistor is connected directly between the two turret terminals. This provides a termination for the coaxial cable at the U29 IC so that the cable is properly matched.

The printed circuit board has been designed such that it will accept a printed circuit type BNC connector. If the user desires, he may purchase a King's KC-79-153 connector for installation directly beneath the pulse generator turret terminals. This will provide a second method for connecting the pulse generator to the Am2900 Evaluation and Learning Kit. In some laboratory environments, this may be more convenient than directly connecting a coaxial cable to the printed circuit board.

## Programming Exercise #1 - Loading the Am2901 Memory

Many of the operations performed with the Am2901 Microprocessor will involve the use of the 16-word, two-address RAM. These 16 RAM locations may be employed at the option of the user as program counters, accumulators, index registers, scratch pad memory, stack pointers, non-user registers, and so forth. Application of the RAM can be determined by the system architecture as well as the microprograms. The purpose of this exercise is to demonstrate one technique for loading the 16 registers from the incoming data input. The exercise is written so as to load a selected word in the RAM memory and then advance to the next line of microprogram code. At this word, the contents of the Am2901 RAM is read out so that it can be examined at the output of the Am2901.

In this exercise, microprogram word zero contains a microinstruction that will load Register "0" of the Am2901 with the value binary two. The microinstruction at word one in the microprogram memory reads the contents of register zero to the Am2901 output via the B data output of the 16 x 4 RAM. Microinstructions 2 and 3 perform the same function on RAM Register "1" while microinstructions 4 and 5 perform the same function on RAM Register "9". Obviously, the user can load and examine all 16 registers in the Am2901 RAM using a similar pair of microinstructions. Incidentally, microinstructions 0 and 1 could be used to perform this function on all 16 registers in the Am2901 by simply re-loading the contents of the microprogram memory "D" and "B" fields and then executing these two instructions; then reloading and executing again.

The actual execution of this exercise is as described below. Leave the RUN/LOAD switch in the LOAD position and set the MEMORY ADDRESS switches to decimal 0. First, depress the SINGLE STEP CLOCK momentary switch so as to enter the contents of the microprogram memory into the pipeline register. Next, advance the MEMORY ADDRESS switches to the decimal 1 position and again depress the SINGLE STEP CLOCK momentary switch. This has the effect of executing the contents of the pipeline register which is currently word zero and entering the contents of microprogram memory word 1 into the pipeline register. The result is that binary two is loaded into Register 0 and the pipeline register contains an instruction that is currently reading the contents of Register 0 to the Am2901 outputs. The user can view the Am2901 outputs on the DATA DISPLAY LED's by setting the RAM & MUX SELECT switches to the decimal 1 position. When this function is performed, the DATA DISPLAY reads binary two (0010).

The next pair of microinstructions which load Register 1 can be executed in a similar fashion. That is, set the MEMORY ADDRESS select switches to the decimal 2 position and depress the SINGLE STEP CLOCK momentary switch. This loads the contents of microprogram memory word 2 into the pipeline register. Next, advance the MEMORY ADDRESS switches to the decimal 3 position and again depress the SINGLE STEP CLOCK momentary switch. This executes the contents of the pipeline register and enters the "read" microinstruction into



the pipeline register. Thus, the user now views the word that has been loaded into Register 1 of the Am2901 RAM. Microinstruction words 4 and 5 are executed in a similar fashion.

The purpose of this exercise is to demonstrate the simple loading of data into the Am2901 memory. The loading of the Q register can be accomplished in exactly the same fashion with only the destination select field of the microinstruction control word being changed. In this example, each "load" instruction has been followed by a "read" instruction so the contents of the Am2901 memory can be examined. In normal operation, only the load operation is performed and the machine moves on to the next task at hand.

Am2900 KIT PROGRAMMING WORK SHEET

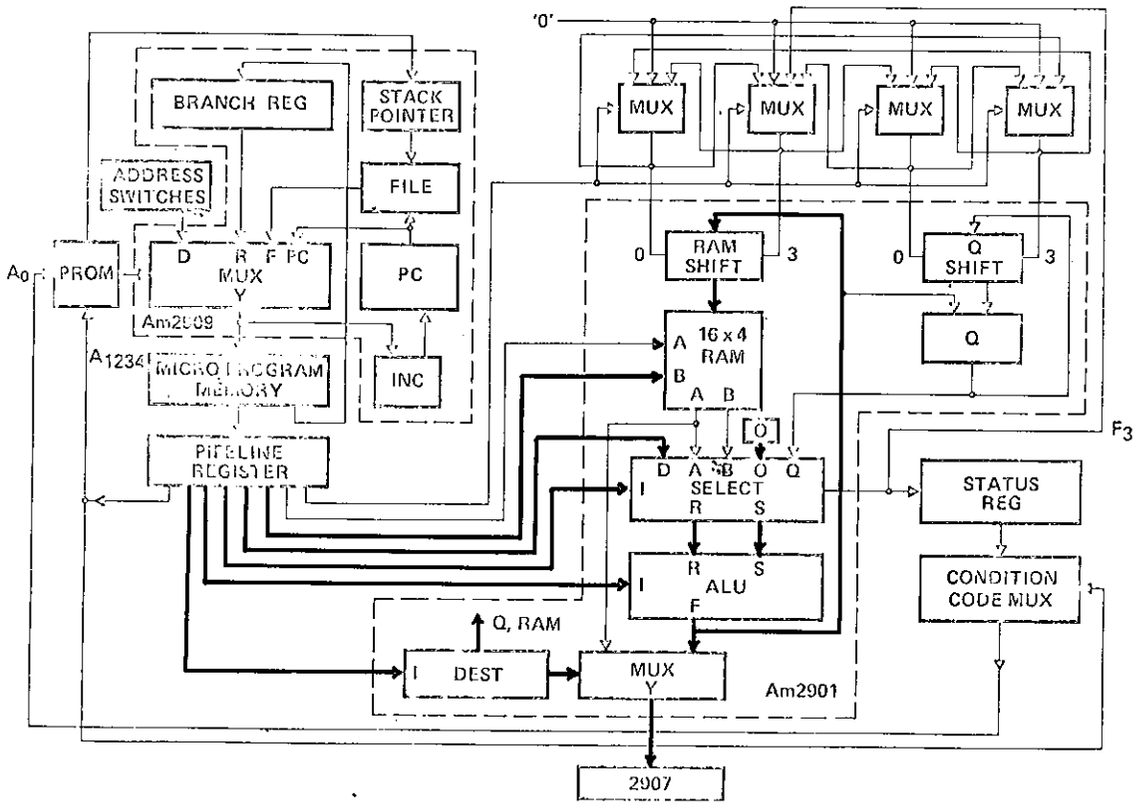
EXERCISE NUMBER: 1 NAME: LOADING THE AM2901 RAM

MICROPROGRAM MEMORY

| RAM & MUX | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | NOTES | |
|---|---|---|---|---|---|---|---|---|---|---|
| | BRANCH ADDRESS | NEXT INSTRUCTION CONTROL | DEST. SELECT | SOURCE SELECT | ALU | 'A' | 'B' | 'D' | NEXT ADDRESS CONTROL | DATA CONTROL |
| MEMORY ADDRESS | $BR_3$ $BR_2$ $BR_1$ $BR_0$ | $P_3$ $P_2$ $P_1$ $P_0$ | MUX1 $I_8$ $I_7$ $I_6$ | MUX0 $I_2$ $I_1$ $I_0$ | $C_n$ $I_5$ $I_4$ $I_3$ | $A_3$ $A_2$ $A_1$ $A_0$ | $B_3$ $B_2$ $B_1$ $B_0$ | $D_3$ $D_2$ $D_1$ $D_0$ | | |
| 0 | | | 0 1 1 | 1 1 1 | 0 1 1 | | 0 0 0 0 | 0 0 1 0 | | LOAD $R_0$ |
| 1 | | | 0 0 1 | 0 1 1 | 0 1 1 | | 0 0 0 0 | | | READ $R_0$ |
| 2 | | | 0 1 1 | 1 1 1 | 0 1 1 | | 0 0 0 1 | 0 1 0 0 | | LOAD $R_1$ |
| 3 | | | 0 0 1 | 0 1 1 | 0 1 1 | | 0 0 0 1 | | | READ $R_1$ |
| 4 | | | 0 1 1 | 1 1 1 | 0 1 1 | | 1 0 0 1 | 0 1 0 1 | | LOAD $R_9$ |
| 5 | | | 0 0 1 | 0 1 1 | 0 1 1 | | 1 0 0 1 | | | READ $R_9$ |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |
| 11 | | | | | | | | | | |
| 12 | | | | | | | | | | |
| 13 | | | | | | | | | | |
| 14 | | | | | | | | | | |
| 15 | | | | | | | | | | |

BLANK - DON'T CARE

INSTRUCTION LOAD $R_1$






## Programming Exercise #2 - Rotate Functions in the Am2901

The Am2900 Evaluation Kit is designed to provide four different shift matrix multiplexer control functions as shown on page 3-9 of the operational description. These four functions are enter zeros, rotate, double length rotate, and double length arithmetic rotate. This shift function can be performed in either the shift-up mode or shift-down mode. The purpose of this exercise is to demonstrate the use of the single length rotate up and down, as well as the double length rotate up and down.

First, the microprogram memory should be loaded with the various data as shown in the accompanying worksheet. Microprogram word 0 is used to load a binary one in the second bit position of Register 0 in the Am2901 RAM. The microinstruction at word 1 performs a rotate up function on this bit. Microprogram memory word 2 performs the rotate down function; word 3 is the double length rotate up function and word 4 is the double length rotate down function. Word 5 is shown to demonstrate the "no op" function.

Exercise 2 should be performed in the following manner. Keep the RUN/LOAD switch in the LOAD position. Set the MEMORY ADDRESS switches to the decimal 0 position and depress the SINGLE STEP CLOCK momentary switch. This will enter the contents of microprogram memory word 0 into the pipeline register. Next, advance the MEMORY ADDRESS switches to the decimal 1 positon. Depress the SINGLE STEP CLOCK momentary switch. If the RAM & MUX SELECT switches are set to the decimal 1 position, the output of the Am2901 Microprocessor will be viewed on the DATA DISPLAY LED's. The current contents of the DATA DISPLAY should be binary two (0010). Now, if the MEMORY ADDRESS switches are left at the decimal 1 positon and the SINGLE STEP CLOCK momentary switch is repeatedly depressed, the bit as displayed on the DATA DISPLAY will be rotated in the upward position. That is, the bit will move in the 2, 4, 8, 1, 2, etc. pattern (0010, 0100, 1000, 0001, 0010, etc.). As many times as the SINGLE STEP CLOCK momentary switch is depressed, the single length rotate up function will be executed.

Next, set the MEMORY ADDRESS switches to the decimal 0 position and depress the SINGLE STEP CLOCK momentary switch. This will again load the contents of microprogram memory word 0 into the pipeline register. Then, set the MEMORY ADDRESS switches to the decimal 2 position and depress the SINGLE STEP CLOCK momentary switch. This will execute the load two microinstruction and enter microprogram memory word 2 into the pipeline register. Now, with the RAM & MUX SELECT switches in the decimal 2 position, the DATA DISPLAY LED's will again show binary two (0010). Keeping the MEMORY ADDRESS switches in the decimal 2 position, each depression of the SINGLE STEP CLOCK momentary switch will result in the execution of a rotate down microinstruction. The pattern on the DATA DISPLAY will follow a 2, 1, 8, 4, 2, etc. sequence as the SINGLE STEP CLOCK momentary switch is depressed.

The double length rotate up and double length rotate down instructions are executed in a similar fashion. First, execute the word at decimal 0 position, then execute the word at the decimal 3 position and then execute the word at the decimal 4 or decimal 5 position. The word at the decimal 3 position simply performs a clear Q function so that the contents of the Q register intially is all zeros. The double length rotate up pattern will now be 2, 4, 8, 0, 0, 0, 0, 1, 2, 4, 8, 0, etc. Likewide, the double length rotate down pattern will be 2, 1, 0, 0, 0, 0, 8, 4, 2, 1, 0, etc.

The "no-operation" function at microprogram memory word 6, if selected, simply demonstrates that the Am2901 can be clocked with no write operation being performed. During any of the above exercises, the MEMORY ADDRESS switches can be changed to the decimal 6 position and the SINGLE STEP CLOCK momemtary switch depressed. The result is that the no-operation function will be entered into the pipeline register and then executed. It should be remembered that when the MEMORY ADDRESS switches are changed to the decimal 6 position, the pipeline register will currently contain a rotate instruction. This instruction will be executed as the contents of the microprogram memory (no-operation) are loaded into the pipeline register. Thus, one additional execution of the rotate function takes place before the no operation instruction is executed.

The purpose of this exercise is to demonstrate how an external multiplexer can be used to perform single length and double length rotates in either direction. The user is encouraged to write instructions that perform shifting of zeros into the word and double length arithmetic shifts. From this example, it should be apparent to the user that a multiplexer scheme can be designed to include the carry flip-flop such that single length and double length rotate with or without carry could be implemented. Likewise, shifting ones or shifting zeros into the Am2901 would be possible.



Am2900 KIT PROGRAMMING WORK SHEET

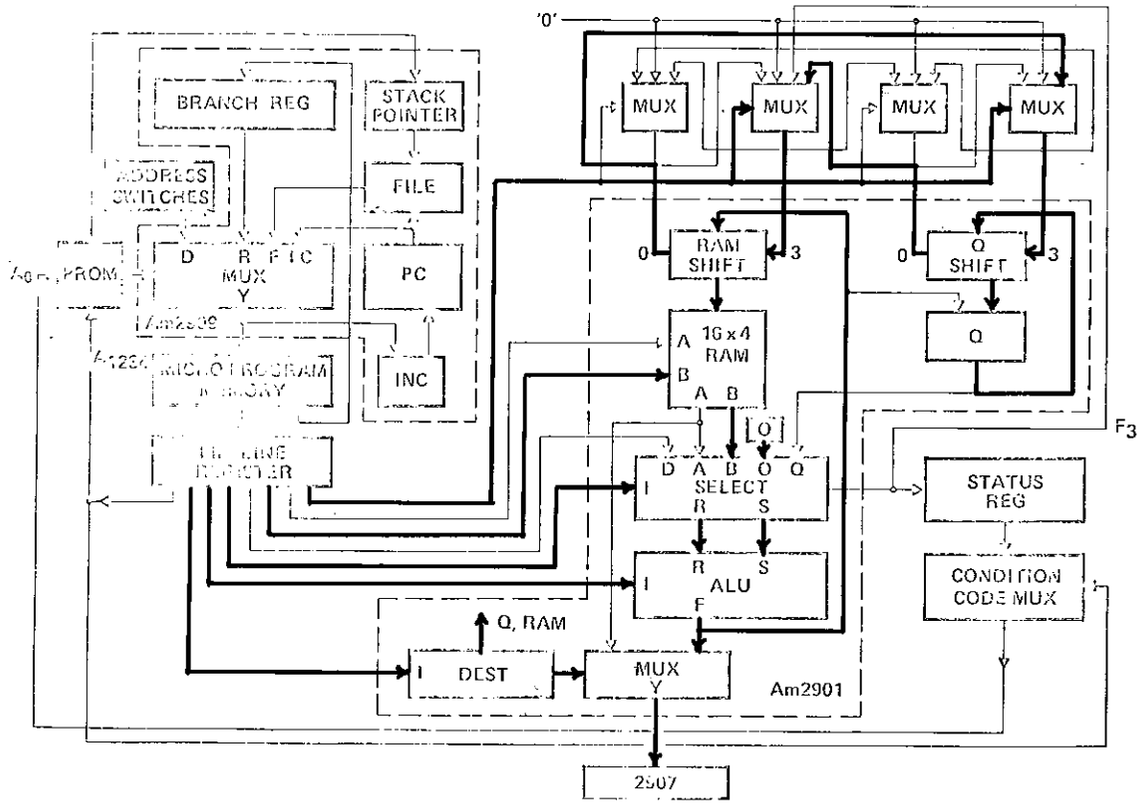
EXERCISE NUMBER: 2 NAME: ROTATE EXERCISES

MICROPROGRAM MEMORY

| MEMORY ADDRESS | 7 PLANNED ADDRESS | 6 TEST INSTRUCTION CONTROL | 5 DEST SELECT | 4 SOURCE SELECT | 3 ALU | 2 "A" | 1 "B" | 0 "D" | NOTES: NEXT ADDRESS CONTROL | NOTES: DATA CONTROL |
|---|---|---|---|---|---|---|---|---|---|---|
| | $BR_3$ $BR_2$ $BR_1$ $BR_0$ | $P_3$ $P_2$ $P_1$ $P_0$ | (MUX) $I_8$ $I_7$ $I_6$ | $I_2$ $I_1$ $I_0$ | $C_n$ $I_5$ $I_4$ $I_3$ | $A_3$ $A_2$ $A_1$ $A_0$ | $B_3$ $B_2$ $B_1$ $B_0$ | $D_3$ $D_2$ $D_1$ $D_0$ | | |
| 0 | | | 0 1 1 | 1 1 1 | 0 1 1 | | 0 0 0 0 | 0 0 1 0 | | LOAD $R_0$ |
| 1 | | | 0 1 1 1 | 1 0 1 | 0 1 1 | | 0 0 0 0 | | | ROT UP |
| 2 | | | 0 1 0 1 | 1 0 1 | 0 1 1 | | 0 0 0 0 | | | ROT DN |
| 3 | | | 0 0 0 | 0 1 0 | 1 0 0 | | | | | CLR Q |
| 4 | | | 1 1 1 0 | 0 0 1 | 0 1 1 | | 0 0 0 0 | | | DBL ROT UP |
| 5 | | | 1 1 0 0 | 0 0 1 | 0 1 1 | | 0 0 0 0 | | | DBL ROT DN |
| 6 | | | 0 0 1 | 0 1 1 | 0 1 1 | | | | | NO OP |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |
| 11 | | | | | | | | | | |
| 12 | | | | | | | | | | |
| 13 | | | | | | | | | | |
| 14 | | | | | | | | | | |
| 15 | | | | | | | | | | |

BLANK = DON'T CARE

INSTRUCTION DOUBLE LENGTH ROTATE DOWN



## Programing Exercise #3 - Am2901 Arithmetic Operation

The purpose of this exercise is to demonstrate a small set of the arithmetic capability of the Am2901 Microprocessor. Four different instruction types are demonstrated on the worksheet for this exercise. The first three types are demonstrated in microprogram memory words 0, 1, and 2 while the fourth type is demonstrated using three microinstructions at memory locations 7, 8, and 9. Again, all data shown in the worksheet should be entered into the microprogram memory.

This exercise is executed in the following manner. Set the RUN/LOAD switch to the LOAD position and set the MEMORY ADDRESS switches to the decimal 0 position. If the RAM & MUX SELECT switches are set to the decimal 1 position, the output of the Am2901 Microprocessor can be viewed on the DATA DISPLAY LED's. Each time the SINGLE STEP CLOCK momentary switch is depressed, the current DATA DISPLAY will be incremented by one.

This is accomplished by using the B and 0 source operands, adding in the ALU with the carry-in set to a one, and writing the results into the Register 0 of the RAM.

If the MEMORY ADDRESS switches are placed in the decimal 1 position, each time the SINGLE STEP CLOCK momentary switch is deprssed, the contents of Register 0 will be decremented. Thus, the DATA DISPLAY will show this decrement function on the contents of Register 0.

If the MEMORY ADDRESS switches are placed at the decimal 2 position, each depression of the SINGLE STEP CLOCK momentary switch will result in the contents of Register 0 being increased by three. Needless to say, an overflow will occur every six clock cycles and the contents of the register will "end around" from a positive value to a negative value. This microinstruction demonstrates adding a data bus input value on the D inputs to the contents of a register.

Microinstruction words 7 and 8 perform the required set-up to demonstrate a register-to-register add operation at microinstruction word 9. This sequence of microinstructions is executed in the following manner. Set the MEMORY ADDRESS switches to the decimal 7 position and depress the SINGLE STEP CLOCK momentary switch. Next, set the MEMORY ADDRESS switches to the decimal 8 position and again depress the SINGLE STEP CLOCK momentary switch. Now, set the MEMORY ADDRESS switches to the decimal 9 position and again depress the SINGLE STEP CLOCK momentary switch. Thus far in the procedure, we have executed microinstruction 8 which sets register 1 to the value five. If we leave the MEMORY ADDRESS switches in the decimal 9 position and begin depressing the SINGLE STEP CLOCK momentary switch, the contents of Register 0 will be increased by the value five. This will be equivalent to a sequence of 0, 5, 10, 15, 4, 9, 14, 3, 8, etc., in a



"magnitude only" number system. This output is viewed on the DATA DISPLAY LED's when the RAM & MUX SELECT switches are in the decimal 1 position. This microinstruction represents a register-to-register arithmetic addition within the Am2901 Microprocessor.

Am2900 KIT PROGRAMMING WORK SHEET

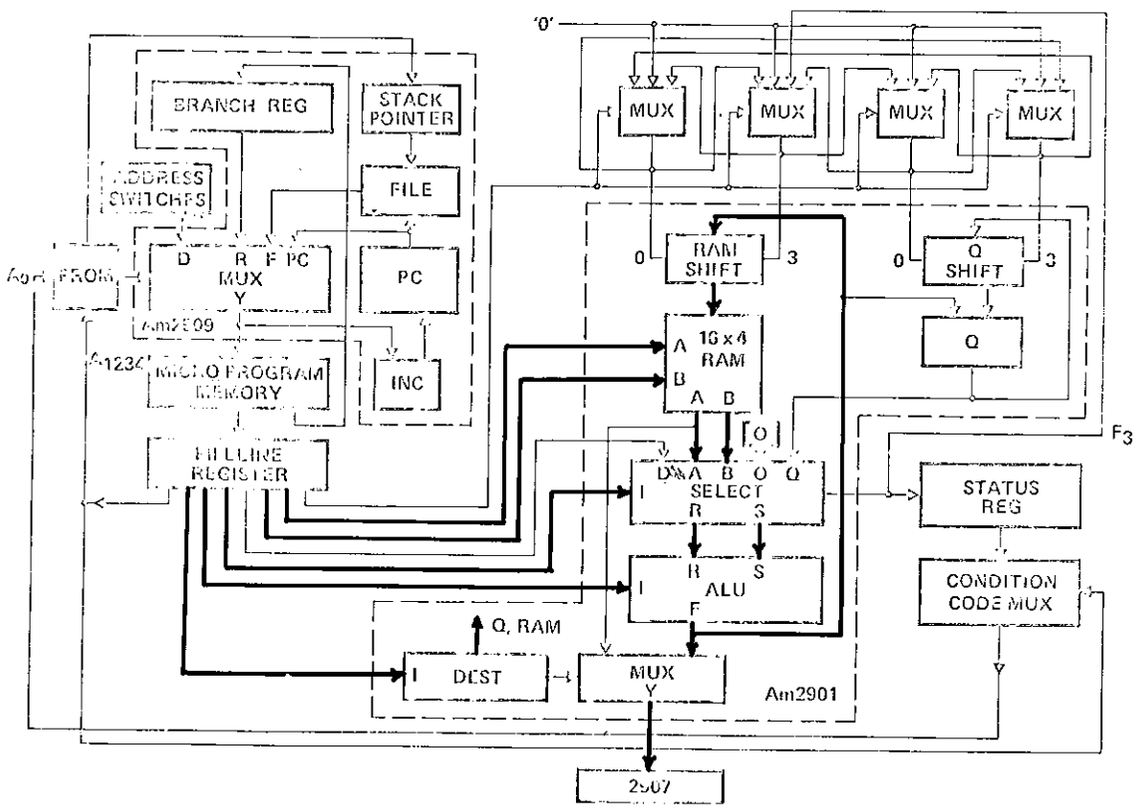
EXERCISE NUMBER: 3 NAME: AM2901 ARITHMETIC OPERATIONS

MICROPROGRAM MEMORY

| MEMORY ADDRESS | 7 BRANCH ADDRESS | 6 NEXT INSTRUCTION CONTROL | 5 DEST. SELECT | 4 SOURCE SELECT | 3 ALU | 2 'A' | 1 'B' | 0 'D' | NOTES |
|---|---|---|---|---|---|---|---|---|---|
| | | $P_3$ $P_2$ $P_1$ $P_0$ | MUX $I_8$ $I_7$ $I_6$ | MUX $I_2$ $I_1$ $I_0$ | $C_n$ $I_5$ $I_4$ $I_3$ | $A_3$ $A_2$ $A_1$ $A_0$ | $B_3$ $B_2$ $B_1$ $B_0$ | $D_3$ $D_2$ $D_1$ $D_0$ | NEXT ADDRESS CONTROL / DATA CONTROL |
| 0 | | | 0 1 1 | 0 1 1 | 1 0 0 0 | | 0 0 0 0 | | INC $R_0$ |
| 1 | | | 0 1 1 | 0 1 1 | 0 0 0 1 | | 0 0 0 0 | | DEC $R_0$ |
| 2 | | | 0 1 1 | 1 0 1 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 1 1 | $R_0 + 3$ |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | 0 1 1 | 0 1 1 | 1 0 0 | | 0 0 0 0 | | CLR $R_0$ |
| 8 | | | 0 1 1 | 1 1 1 | 0 1 1 | | 0 0 0 1 | 0 1 0 1 | $R_1 = 5$ |
| 9 | | | 0 1 1 | 0 0 1 | 0 0 0 0 | 0 0 0 1 | 0 0 0 0 | | $R_0 \leftarrow R_0 + R_1$ |
| 10 | | | | | | | | | |
| 11 | | | | | | | | | |
| 12 | | | | | | | | | |
| 13 | | | | | | | | | |
| 14 | | | | | | | | | |
| 15 | | | | | | | | | |

BLANK = DON'T CARE

INSTRUCTION $R_0 \leftarrow R_0 + R_1$






## Programming Exercise #4 - Continue and Branch Next Instructions in the Am2909 Microprogram Sequencer

The next few exercises will examine various functions for the control of the next microinstruction address. Field 6 of the microprogram memory word contains a number of functions that are executed by the next address control PROM associated with the Am2909 Microprogram Sequencer. These functions are shown on page 3-5 of the discussion on the operation of the Am2900 Kit. Exercises 1, 2, and 3 demonstrated a few of the functions of the Am2901 Microprocessor. All of these microinstructions were executed with the RUN/LOAD switch in the LOAD position. In order to execute the microinstructions associated with the next address control of the Am2909 Microprogram Sequencer, it will be necessary to switch the RUN/LOAD switch to the RUN position. However, in order to load the microprogram memory, the RUN/LOAD switch must be in the LOAD position.

First, the data on the programming table of the exercise 4 worksheet should be loaded into the microprogram memory. This exercise demonstrates the CONTINUE (or EXECUTE) operation as well as the BRANCH (or JUMP) operation in next microinstruction select control. After the microprogram memory has been loaded, the MEMORY ADDRESS switches should be placed in the decimal 0 position and the SINGLE STEP CLOCK momentary switch depressed one time. This will load the contents of memory address zero into the pipeline register. This is necessary to initialize the pipeline register so that the instruction sequence in the Exercise 4 worksheet can be executed. If this step is not performed, the current contents of the pipeline register are not known and the next microprogram memory address will most likely not be as determined from the worksheet. Now, the RUN/LOAD switch should be changed to the RUN position. If the RAM & MUX SELECT switches are placed in the decimal 0 position, the output of the Am2909 Microprogram Sequencer can be viewed on the DATA DISPLAY LED's. This display represents the address of the next microinstruction. At this point in the execution of the exercise, the DATA DISPLAY should show decimal nine (1001). As the SINGLE STEP CLOCK momentary switch is depressed, the sequence on the DATA DISPLAY should be 9, 6, 10, 11, 12, 13, 14, 15, 3, 0, 9, 6, etc. Microinstruction at address 0, 3, 6, 9, and 15 are BRANCH instructions while microinstructions at addresses 10, 11, 12, 13, and 14 are CONTINUE instructions.

The purpose of this exercise, of course, is to demonstrate the simple CONTINUE and BRANCH instructions used throughout microprogram memory next address control.

Am2900 KIT PROGRAMMING WORK SHEET

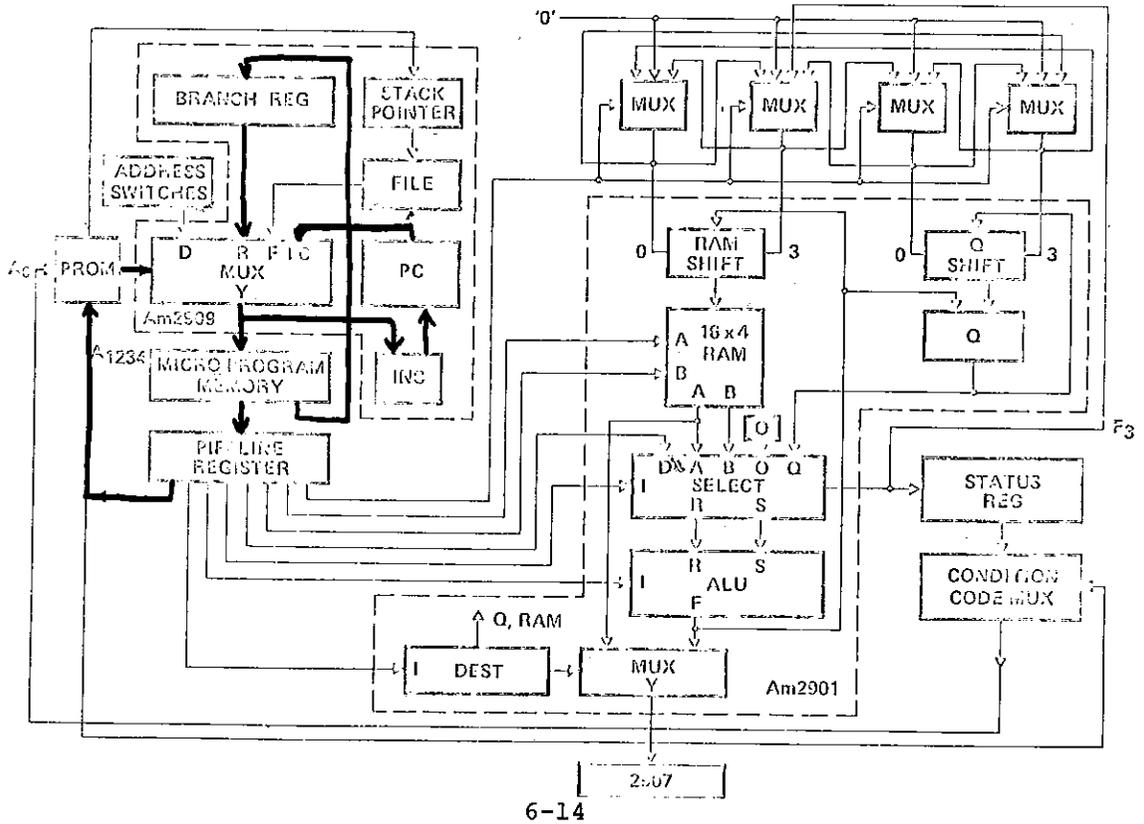
EXERCISE NUMBER: 4 NAME: CONTINUE AND BRANCH

MICROPROGRAM MEMORY

| MEMORY ADDRESS | RAM & MUX | 7 BRANCH ADDRESS $BR_3$ $BR_2$ $BR_1$ $BR_0$ | 6 NEXT INSTRUCTION CONTROL $P_3$ $P_2$ $P_1$ $P_0$ | 5 MUX | DEST. SELECT $I_8$ $I_7$ $I_6$ | 4 MUX | SOURCE SELECT $I_2$ $I_1$ $I_0$ | 3 $C_n$ | ALU $I_5$ $I_4$ $I_3$ | 2 "A" $A_3$ $A_2$ $A_1$ $A_0$ | 1 "B" $B_3$ $B_2$ $B_1$ $B_0$ | 0 "D" $D_3$ $D_2$ $D_1$ $D_0$ | NOTES: NEXT ADDRESS CONTROL | NOTES: DATA CONTROL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 1001 | 0001 | | | | | | | | | | BR 9 | |
| 1 | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | |
| 3 | | 0000 | 0001 | | | | | | | | | | BR 0 | |
| 4 | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | |
| 6 | | 1010 | 0001 | | | | | | | | | | BR 10 | |
| 7 | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | |
| 9 | | 0110 | 0001 | | | | | | | | | | BR 6 | |
| 10 | | | 0010 | | | | | | | | | | CONT | |
| 11 | | | 0010 | | | | | | | | | | CONT | |
| 12 | | | 0010 | | | | | | | | | | CONT | |
| 13 | | | 0010 | | | | | | | | | | CONT | |
| 14 | | | 0010 | | | | | | | | | | CONT | |
| 15 | | 0011 | 0001 | | | | | | | | | | BR 3 | |

BLANK = DON'T CARE

INSTRUCTION SEQUENCE: 0, 9, 6, 10, 11, 12, 13, 14, 15 3, 0, ETC






## Programming Exercise #5 - Looping in Microprogram Memory

The purpose of this exercise is simply to demonstrate the technique of looping within the microprogram memory. This particular demonstration does not make provision for braching out of the loop in any fashion. While this is not the normal case, it does provide the student with a view of what is involved in getting into and executing a loop. The loop is normally terminated by using one of two types of instructions. The first type of instruction would be test-end-of-loop and either repeat the loop if the condition is not true or continue out of the loop if the condition is true. When the "continue out of the loop" microinstruction is performed, a POP is executed to keep the stack maintained properly. The second technique for escaping a loop is to perform a conditional branch microinstruction somewhere within one of the loop microinstructions and when the test condition finally becomes true, a branch from the loop is made. Again, once out of the loop the first microinstruction should be a POP to perform the file maintenance.

Once the data on the worksheet associated with Exercise 5 has been loaded in the microprogram memory, the MEMORY ADDRESS select switches should be placed at decimal 0 and the SINGLE STEP CLOCK momentary switch depressed. Now, the RUN/LOAD select switch should be changed to the RUN position. If the RAM & MUX SELECT switches are placed in the decimal 0 position, the DATA DISPLAY LED's will view the Am2909 Microprogram Sequencer next address output. At this point, the DATA DISPLAY will show decimal 1. As the SINGLE STEP CLOCK momentary switch is depressed, the DATA DISPLAY LED's sequence will follow the pattern of 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 3, 4, 5, etc. Thus, the microinstruction 3 and proceeding through microinstruction 12, at which point a loop back to microinstruction 3 is performed. This loop is made possible by using the file reference next address microinstruction. A PUSH was performed at microword 2 so that the word currently on the stack is address 3. Thus, each time the file reference instruction is executed at microword 12, the next address from the Am2909 Microprogram Sequencer is microinstruction address 3.

Am2900 KIT PROGRAMMING WORK SHEET

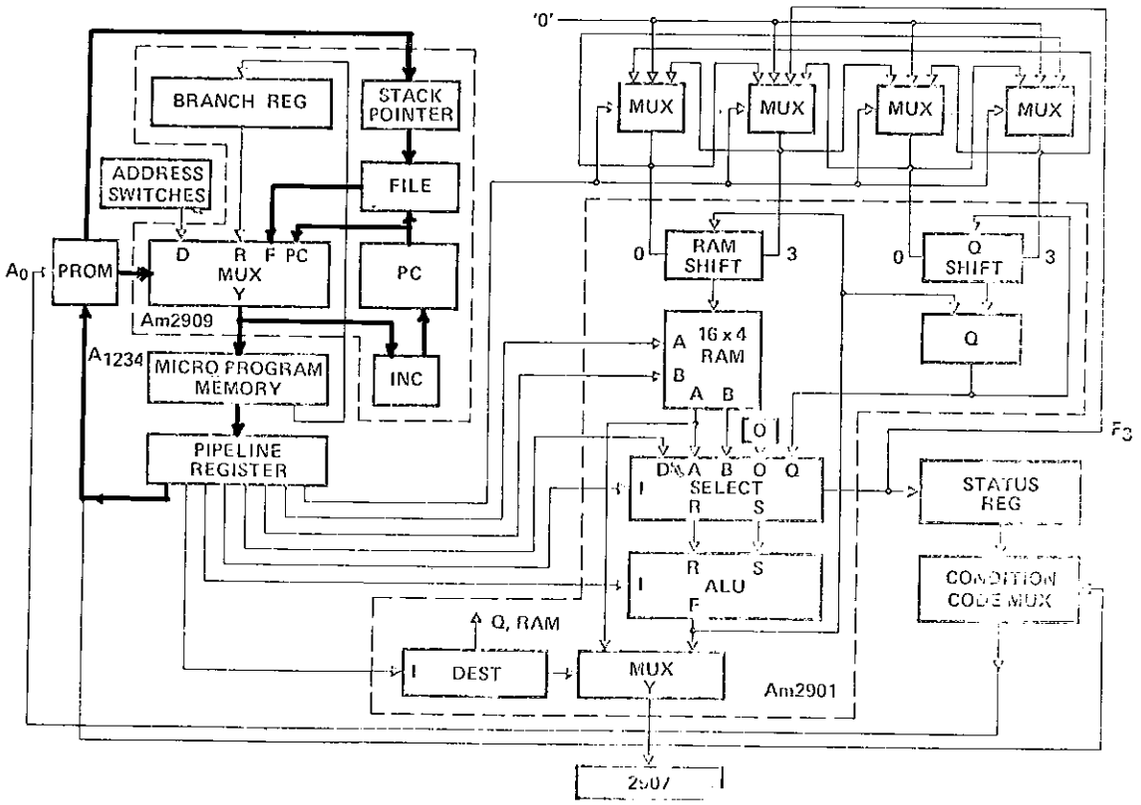
EXERCISE NUMBER: 5 NAME: LOOPING IN MICRO PROGRAM MEMORY

MICROPROGRAM MEMORY

| RAM & MUX | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | NOTES | |
|---|---|---|---|---|---|---|---|---|---|---|
| | BRANCH ADDRESS | NEXT INSTRUCTION CONTROL | DEST. SELECT | SOURCE SELECT | ALU | 'A' | 'B' | 'D' | | |
| MEMORY ADDRESS | $BR_3$ $BR_2$ $BR_1$ $BR_0$ | $P_3$ $P_2$ $P_1$ $P_0$ | MUX1 $I_8$ $I_7$ $I_6$ | MUX0 $I_2$ $I_1$ $I_0$ | $C_n$ $I_5$ $I_4$ $I_3$ | $A_3$ $A_2$ $A_1$ $A_0$ | $B_3$ $B_2$ $B_1$ $B_0$ | $D_3$ $D_2$ $D_1$ $D_0$ | NEXT ADDRESS CONTROL | DATA CONTROL |
| 0 | | 0010 | | | | | | | CONT | |
| 1 | | 0010 | | | | | | | CONT | |
| 2 | | 1001 | | | | | | | PUSH | |
| 3 | | 0010 | | | | | | | CONT | |
| 4 | | 0010 | | | | | | | CONT | |
| 5 | | 0010 | | | | | | | CONT | |
| 6 | | 0010 | | | | | | | CONT | |
| 7 | | 0010 | | | | | | | CONT | |
| 8 | | 0010 | | | | | | | CONT | |
| 9 | | 0010 | | | | | | | CONT | |
| 10 | | 0010 | | | | | | | CONT | |
| 11 | | 0010 | | | | | | | CONT | |
| 12 | | 0111 | | | | | | | FILE REF | |
| 13 | | | | | | | | | | |
| 14 | | | | | | | | | | |
| 15 | | | | | | | | | | |

BLANK = DON'T CARE

INSTRUCTION SEQUENCE: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 3, 4, ETC




6-16

Programming Exercise #6 - Executing a Subroutine in Microprogram Control

The purpose of this exercise is to demonstrate the technique of subroutining in microprogram memory. Microinstruction 3 executes a jump-to-subroutine at microinstruction 12. The subroutine at microinstruction 12 is three microinstructions in length covering the microprogram memory space between microinstruction 12 and microinstruction 14. The basic microinstruction sequence between microword 0 and microword 6 is simply a continue sequence between 0 and 3 as well as from 4 to 6. At microword 6, a branch to word 0 is performed.

This exercise is executed by first loading the microprogram memory with the data on the worksheet. Once the microprogram memory has been loaded, the MEMORY ADDRESS select switches should be placed in the decimal 0 position and the SINGLE STEP CLOCK momentary switch should be depressed. Now, the RUN/LOAD select switch should be placed to the RUN position and the RAM & MUX SELECT switches placed to the decimal 0 position. This alllows the DATA DISPLAY LED's to view the output of the Am2909 Microprogram Sequencer. The current display should be decimal value 1. As the SINGLE STEP CLOCK momentary switch is deprssed, the DATA DISPLAY should execute the sequence 1, 2, 3, 12, 13, 14, 4, 5, 6, 0, 1, 2, 3, 12, etc.

6-17

Am2900 KIT PROGRAMMING WORK SHEET

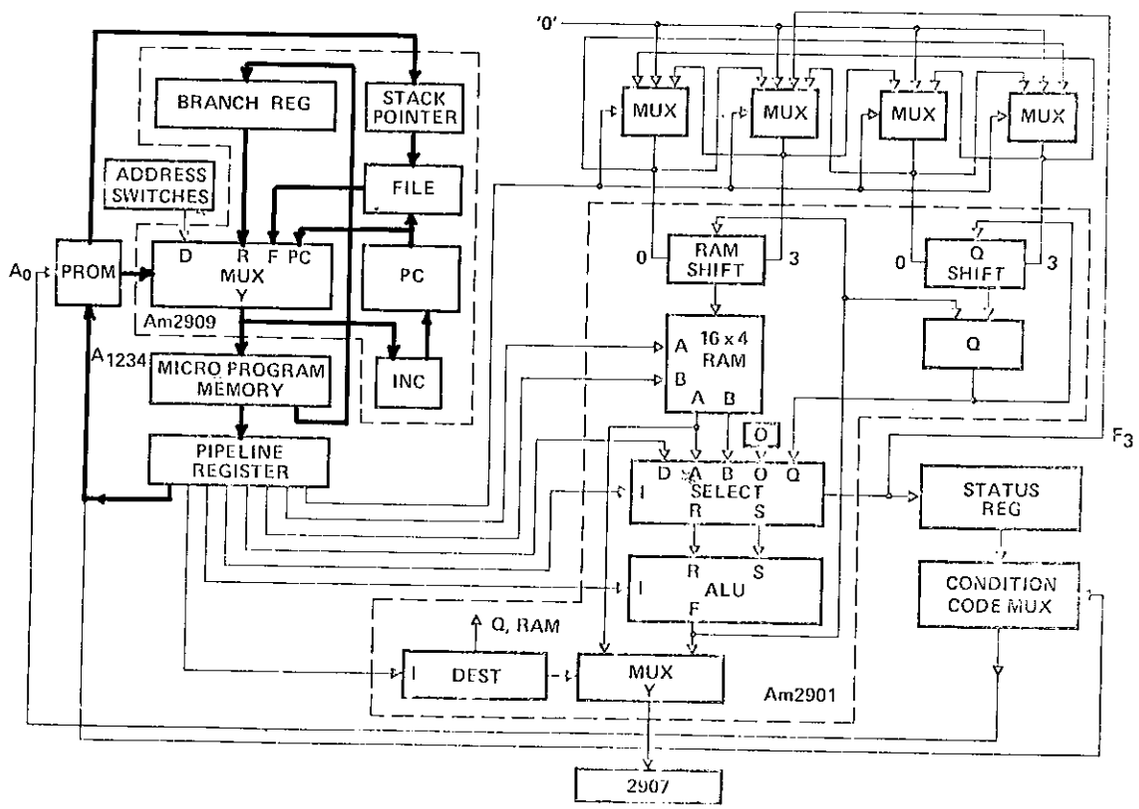
EXERCISE NUMBER: 6 NAME: JUMP TO ONE SUBROUTINE

MICROPROGRAM MEMORY

| RAM & MUX | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | NOTES | |
|---|---|---|---|---|---|---|---|---|---|---|
| | BRANCH ADDRESS | NEXT μINSTRUCTION CONTROL | DEST. SELECT | SOURCE SELECT | ALU | "A" | "B" | "D" | | |
| MEMORY ADDRESS | $BR_3$ $BR_2$ $BR_1$ $BR_0$ | $P_3$ $P_2$ $P_1$ $P_0$ | MUX$_A$ $I_8$ $I_7$ $I_6$ | MUX$_B$ $I_2$ $I_1$ $I_0$ | $C_n$ $I_5$ $I_4$ $I_3$ | $A_3$ $A_2$ $A_1$ $A_0$ | $B_3$ $B_2$ $B_1$ $B_0$ | $D_3$ $D_2$ $D_1$ $D_0$ | NEXT ADDRESS CONTROL | DATA CONTROL |
| 0 | | 0010 | | | | | | | CONT | |
| 1 | | 0010 | | | | | | | CONT | |
| 2 | | 0010 | | | | | | | CONT | |
| 3 | 1100 | 0101 | | | | | | | JSB 12 | |
| 4 | | 0010 | | | | | | | CONT | |
| 5 | | 0010 | | | | | | | CONT | |
| 6 | 0000 | 0001 | | | | | | | BR 0 | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |
| 11 | | | | | | | | | | |
| 12 | | 0010 | | | | | | | CONT | |
| 13 | | 0010 | | | | | | | CONT | |
| 14 | | 0110 | | | | | | | RTS | |
| 15 | | | | | | | | | | |

BLANK DON'T CARE

INSTRUCTION _____






## Programming Exercise #7 - Nesting Subroutines

This exercise demonstrates the technique of nesting subroutines in microprogram memory. The subroutines are nested four levels deep in this example. The main microprogram resides at microinstructions 13, 14, and 15. The microinstruction at word 14 is a jump-to-subroutine at address 0. This is shown on the flow diagram below. Once at subroutine 0, we see the first instruction is a jump-to-subroutine at microinstruction 12. The subroutine at microinstruction 12 is a single microinstruction subroutine resulting in a return-from-subroutine. The microinstruction at address 1 is another jump-to-subroutine at microprogram address 6. The subroutine beginning at microprogram address 6 again executes a jump-to-subroutine at address 12. Again, the single microinstruction subroutine is executed and the microprogram control returns to address 7 where another jump-to-subroutine is found. Here, the jump-to-subroutine takes the next microinstruction from word 3. The microinstruction at word 3 again executes a jump-to-subroutine at word 12. The single microinstruction subroutine at location 12 is now executed and the microprogram control returns to address 4. The microprogram instruction at word 4 is another jump-to-subroutine at microprogram memory word 9. Microprogram memory word 9 is a continue instruction and microprogram memory word 10 is a return-from-subroutine instruction. The flow is such that subroutine 9 returns to subroutine 3 which returns to subroutine 6 which returns to subroutine 0 which returns to the main program.

From this example, several observations can be made about subroutining in microprogram control. First, we have nested subroutines up to four levels deep using the Am2909 stack. Second, the stack has maintained the correct return linkage throughout the various subroutine calls and returns. Third, the subroutine at microinstruction 12 has been used at more than one level of subroutining.

The purpose of this exercise has been to demonstrate nesting of four levels of subroutining. Note that each level of subroutining required one return address location in the PUSH/POP stack. Recalling the discussion on looping in Exercise 5, a PUSH onto the stack was performed to supply the reference address for the loop. Thus, it should be understood that up to four levels of loops and subroutines can be intermixed in any fashion. One word of the stack is used for each reference address required. For example, a subroutine might contain a loop which contains another subroutine which contains another loop. This would utilize the four levels of stack depth within the Am2909 Microprogram Sequencer.

Am2900 KIT PROGRAMMING WORK SHEET

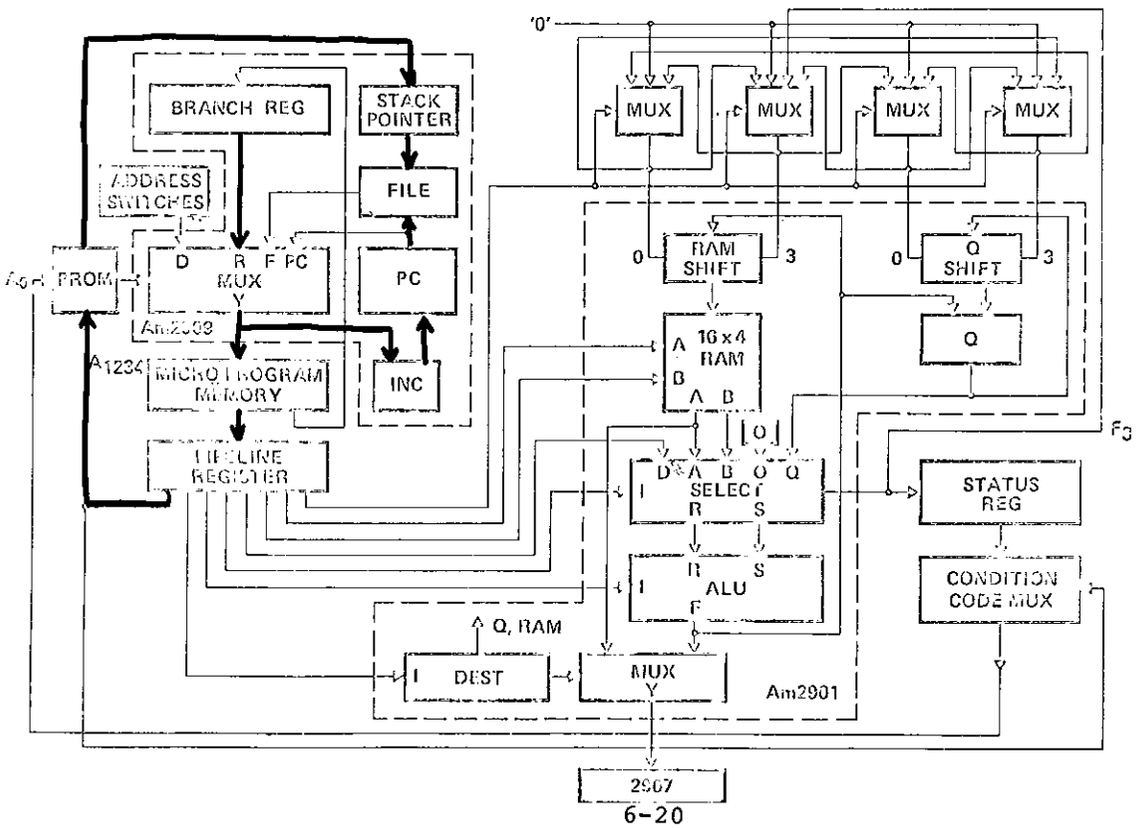
EXERCISE NUMBER: 7 NAME: NESTING SUBROUTINES

MICROPROGRAM MEMORY

| RAM & MUX | 7 | 6 | 5 | | 4 | | 3 | 2 | 1 | 0 | NOTES | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BRANCH ADDRESS | NEXT INSTRUCTION CONTROL | | DEST. SELECT | | SOURCE SELECT | | ALU | 'A' | 'B' | 'D' | NEXT ADDRESS CONTROL | DATA CONTROL |
| MEMORY ADDRESS | $BR_3$ $BR_2$ $BR_1$ $BR_0$ | $P_3$ $P_2$ $P_1$ $P_0$ | MUX1 | $I_8$ $I_7$ $I_6$ | MUX0 | $I_2$ $I_1$ $I_0$ | $C_n$ | $I_5$ $I_4$ $I_3$ | $A_3$ $A_2$ $A_1$ $A_0$ | $B_3$ $B_2$ $B_1$ $B_0$ | $D_3$ $D_2$ $D_1$ $D_0$ | | |
| 0 | 1100 | 0111 | | | | | | | | | | JSB 12 | |
| 1 | 0110 | 0111 | | | | | | | | | | JSB 6 | |
| 2 | | 0101 | | | | | | | | | | RTS | |
| 3 | 1100 | 0111 | | | | | | | | | | JSB12 | |
| 4 | 1001 | 0111 | | | | | | | | | | JSB9 | |
| 5 | | 0101 | | | | | | | | | | RTS | |
| 6 | 1100 | 0111 | | | | | | | | | | JSB 12 | |
| 7 | 0011 | 0111 | | | | | | | | | | JSB 3 | |
| 8 | | 0101 | | | | | | | | | | RTS | |
| 9 | | 0001 | | | | | | | | | | CONT | |
| 10 | | 0101 | | | | | | | | | | RTS | |
| 11 | | | | | | | | | | | | | |
| 12 | | 0101 | | | | | | | | | | RTS | |
| 13 | | 0010 | | | | | | | | | | CONT | |
| 14 | 0000 | 0111 | | | | | | | | | | JSB 0 | |
| 15 | 1101 | 0000 | | | | | | | | | | BR 13 | |

BLANK = DON'T CARE

INSTRUCTION JSB 12






MAIN MICROPROGRAM

| | |
|---|---|
| 13 | CONTINUE |
| 14 | JSB-0 |
| 15 | BRANCH-13 (SYNC) |

SUBROUTINE-12

| | |
|---|---|
| 12 | RTS |

SUBROUTINE-0

| | |
|---|---|
| 0 | JSB-12 |
| 1 | JSB-6 |
| 2 | RTS |

SUBROUTINE-6

| | |
|---|---|
| 6 | JSB-12 |
| 7 | JSB-3 |
| 8 | RTS |

SUBROUTINE-0

| | |
|---|---|
| 3 | JSB-12 |
| 4 | JSB-9 |
| 5 | RTS |

SUBROUTINE-9

| | |
|---|---|
| 9 | CONTINUE |
| 10 | RTS |

Actual Address flow is:

13, 14, 0, 12, 1, 6, 12, 7, 3, 12, 9, 10, 5, 8, 2, 15, 13, etc.

## Programming Exercise #8 - Combining the Am2901 and Am2909 to Perform Conditional Branching

This is the first exercise that combines both the Am2901 and Am2909 functions. It is used to demonstrate the technique of conditional branching in microprogram control. The microinstruction at word 1 contains a test on the carry flag whereby a conditional branch occurs if the carry output is logic 1.

This exercise should be executed in the following manner. After the data on the worksheet has been loaded into the microprogram memory, the MEMORY ADDRESS select switches should be placed at decimal 0 and the SINGLE STEP CLOCK momentary switch depressed. Now, the RUN/LOAD select switch should be placed in the RUN position. If the RAM & MUX select switches are placed in the decimal 0 position, the Am2909 Microprogram Sequencer next microinstruction address output is viewed. The DATA DISPLAY LED's will currently contain decimal value 1. As the SINGLE STEP CLOCK momentary switch is depressed, the data display pattern will follow a 1, 2, 0, 1, 2, 0 pattern until the carry output is a "logic 1". At this point, the next address will branch from microprogram memory word 1 to microprogram memory word 15 and then to microprogram memory word 0. This 1, 2, 0 loop will be repeated 15 times until the next carry flag occurs on the 16th interation and a branch to microword 15 again occurs.

The way this exercise operates is to take the contents of register 0 and increment the current value on microword 0. The instruction at microword 1 is used to test the carry flag stored in the status register. If the carry flag is logic 1, a branch to microword 15 is performed. At microword 15, a continue instruction is executed which results in an end around microword 0.

While this exercise is contained in the static testing mode description, it can also be used in the dynamic testing mode. Microinstruction 15 will occur once every 48 clock cycles. Because the design of the Am2900 Kit uses the Am2909 Sequencer with the carry input tied HIGH, each time address 15 occurs at the Am2909 outputs, the Am2909 carry output will be HIGH. It is this Am2909 carry output that is connected directly to the SYNC TEST turret terminal on the kit printed circuit board. Thus, in this example, the SYNC TEST turret terminal provides a convenient point to synchronize the oscilloscope to see the entire 48 word microprogram sequence. Likewise, the ADDRESS SYNC turret terminal can be used to view particular microinstruction address execution on an oscilloscope. That is, by setting the MEMORY ADDRESS select switches to a particular decimal value, a HIGH output on the ADDRESS SYNC turret terminal results each time the Am2909 Microprogram Sequencer address matches that of the MEMORY ADDRESS select switches. This allows one trace of a multiple trace oscilloscope to be connected to the ADDRESS SYNC turret terminal and the execution of a particular address studied. This will be discussed in more detail in the dynamic operational description of the programming exercises.



Am2900 KIT PROGRAMMING WORK SHEET

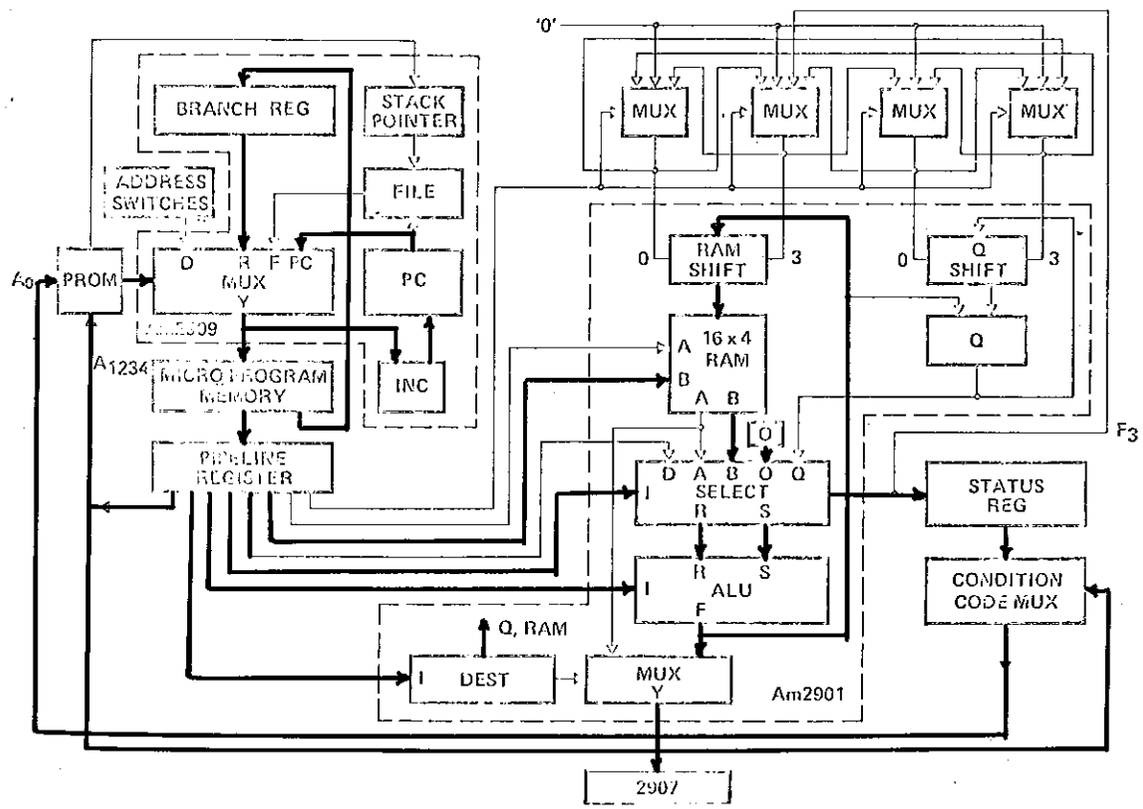
EXERCISE NUMBER: 8 NAME: INCREMENT AND TEST R0

MICROPROGRAM MEMORY

| MEMORY ADDRESS | 7 BRANCH ADDRESS $BR_3$ $BR_2$ $BR_1$ $BR_0$ | 6 NEXT INSTRUCTION CONTROL $I_{11}$ $I_{10}$ $I_9$ $I_8$ | 5 DEST SELECT $I_8$ $I_7$ $I_6$ | 4 SOURCE SELECT $I_2$ $I_1$ $I_0$ | 3 ALU $I_5$ $I_4$ $I_3$ | 2 "A" $A_3$ $A_2$ $A_1$ $A_0$ | 1 "B" $B_3$ $B_2$ $B_1$ $B_0$ | 0 "D" $D_3$ $D_2$ $D_1$ $D_0$ | NOTES: NEXT ADDRESS CONTROL | NOTES: DATA CONTROL |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0010 | 011 | 011 | 1000 | | 0000 | | CONT | INC R0 |
| 1 | 1111 | 1111 | 001 | | | | | | COND BR | BR IS Cn+4 |
| 2 | 0000 | 0001 | 001 | | | | | | BR 0 | NO OP |
| 3 | | | | | | | | | | |
| 4 | | | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |
| 11 | | | | | | | | | | |
| 12 | | | | | | | | | | |
| 13 | | | | | | | | | | |
| 14 | | | | | | | | | | |
| 15 | | 0010 | 001 | | | | | | CONT | NO OP |

BLANK = DON'T CARE

INSTRUCTION _____



## Programming Exercise #9 - Measuring the B Address Access Time

The purpose of this exercise is to demonstrate the technique used to measure the access time of the Am2901 RAM with respect to the B address. First, microprogram memory word 0, 1, 14, and 15 should be loaded as shown on the worksheet. Now, the CLOCK SELECT switch should be in the SINGLE STEP position and the RUN/LOAD select switch should be in the LOAD position. The MEMORY ADDRESS select switches should be in the decimal 0 position and the SINGLE STEP CLOCK momentary switch depressed. This initializes the pipeline register to a starting address. Next, change the RUN/LOAD select switch to the RUN position. Now, depress the SINGLE STEP CLOCK momemtary switch two times so as to execute the microinstructions at word 0 and word 1. This performs the loading of register 0 with all zeros and register 15 with all ones.

Now, if a pulse generator has been connected to the PULSE GENERATOR inputs, when the CLOCK SELECT switch is changed to the PULSE GENERATOR position, the Am2900 Evaluation Kit will be operating in the dynamic mode. The SYNC TEST turret terminal can be used as a convenient oscilloscope sync point. The microprogram control will now be executing the microinstructions at word 14 and word 15.

The two microinstructions at word 14 and word 15 cause the B address to be changed from all zeros to all ones and then back to all zeros. The Am2901 RAM data output for these two registers is also all zeros and all ones. Thus, if the Am2901 Y outputs are probed with an oscilloscope and the Am2901 B inputs are also examined with an oscilloscope, the differential time between the B address change and the Y data output is measured - the B access time. The path selected in this example is to use the ALU "OR" function with the B and O source operands. In this example, the RAM is being re-written on each microinstruction; however, the no operation destination control instruction can also be used.

From this example, the user should recognize that the B address fields and the data fields could be selected in any fashion so that various patterns can be examined. Likewise, the entire Am2901 RAM might be initialized to all ones, all zeros, or a checkerboard pattern if the user desires. It should also be recognized that similar techniques can be used to measure various propagation delays such as the D inputs to the Y outputs, the A address inputs to the Y address outputs, carry-in to carry-out and so forth. In fact, almost all of the combinatorial propagation delays shown in Table II of the Am2901 data sheet can be measured in this manner. The key item to be remembered in performing any of these measurements is that only the variable to be measured changes between microcycles. That is, all other inputs to the Am2901 should be held constant except for the path being measured. For example, when measuring the carry input to carry output propagation delay, only the carry input should be changed during the microinstruction time of interest.



Am2900 KIT PROGRAMMING WORK SHEET

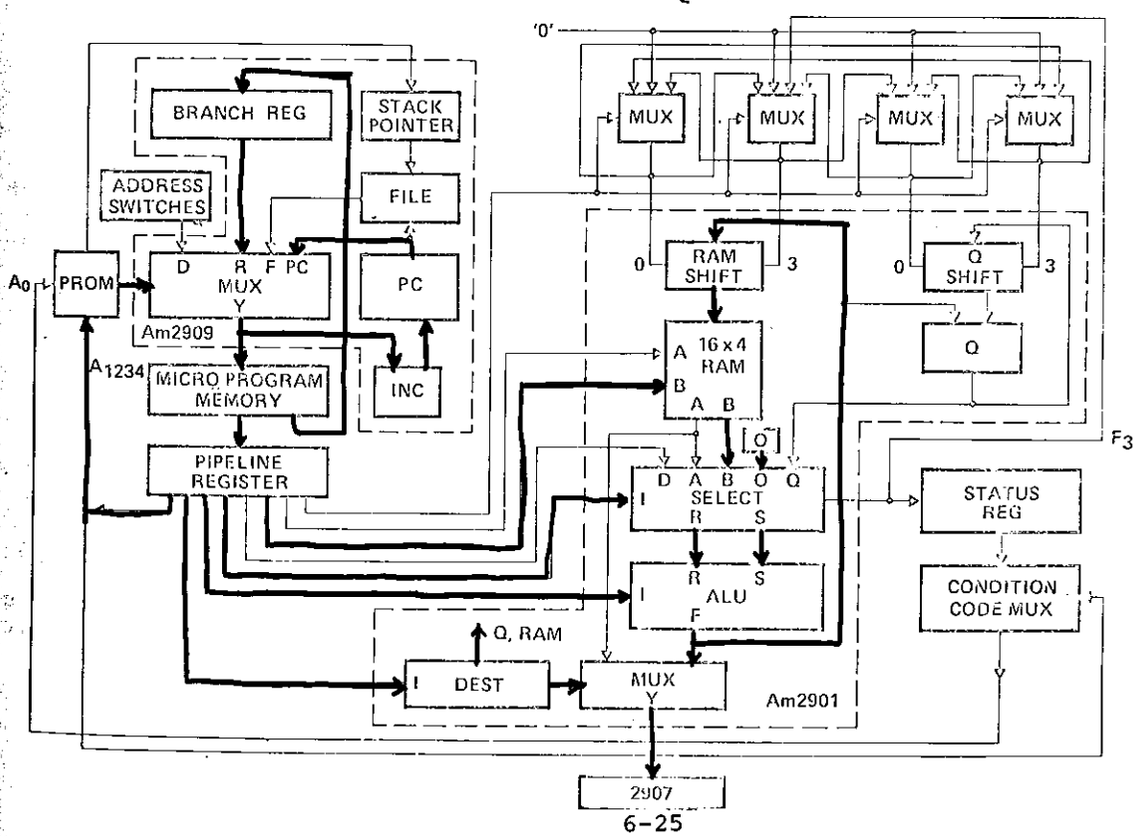
EXERCISE NUMBER: 9 NAME: EXAMPLE OF "B" ACCESS TIME

MICROPROGRAM MEMORY

| RAM & MUX / MEMORY ADDRESS | 7 BRANCH ADDRESS $BR_3$ $BR_2$ $BR_1$ $BR_0$ | 6 NEXT INSTRUCTION CONTROL $P_3$ $P_2$ $P_1$ $P_0$ | 5 DEST. SELECT MUX$_1$ $I_8$ $I_7$ $I_6$ | 4 SOURCE SELECT MUX$_0$ $I_2$ $I_1$ $I_0$ | 3 ALU $C_n$ $I_5$ $I_4$ $I_3$ | 2 "A" $A_3$ $A_2$ $A_1$ $A_0$ | 1 "B" $B_3$ $B_2$ $B_1$ $B_0$ | 0 "D" $D_3$ $D_2$ $D_1$ $D_0$ | NOTES: NEXT ADDRESS CONTROL | NOTES: DATA CONTROL |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0010 | 011 | 111 | 011 | | 0000 | 0000 | CONT | $R_0 = 0$ |
| 1 | 1110 | 0001 | 011 | 111 | 011 | | 1111 | 1111 | BR 14 | $R_{15} = 15$ |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |
| 11 | | | | | | | | | | |
| 12 | | | | | | | | | | |
| 13 | | | | | | | | | | |
| 14 | | 0010 | 011 | 011 | 011 | | 0000 | | CONT | READ $R_0$ |
| 15 | 1110 | 0001 | 011 | 011 | 011 | | 1111 | | BR 14 | READ $R_{15}$ |

BLANK = DON'T CARE

INSTRUCTION B Access (14, 15)



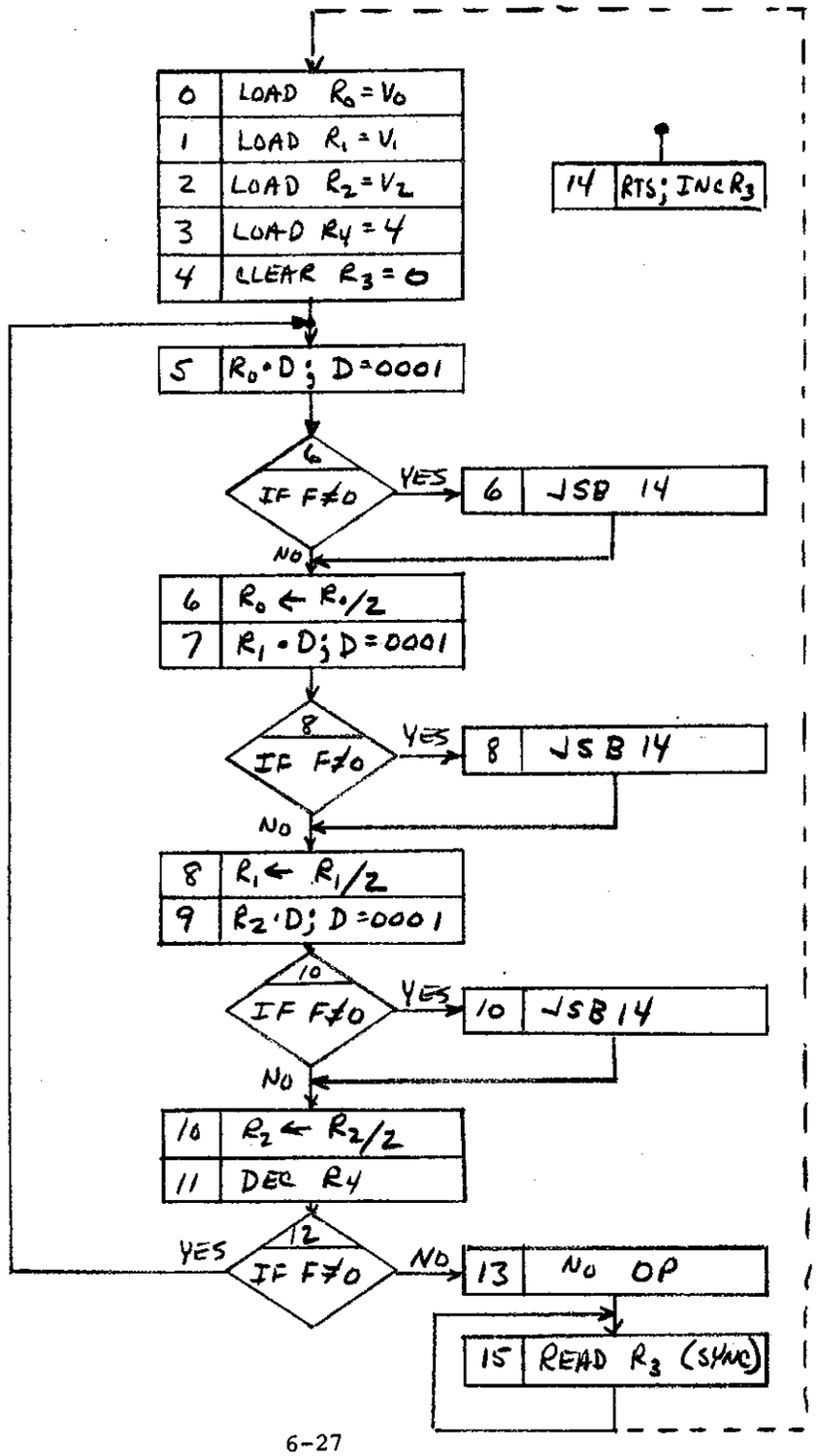
## Programming Exercise #10 - Counting the Total Number of Ones in Three Register Words

This exercise demonstrates the use of the conditional jump-to-subroutine function and uses most of the paths in the Am2900 Evaluation and Learning Kit. $V_0$, $V_1$, and $V_2$ are three data values that are written into the data field of microprogram memory words 0, 1, and 2. The user should make up these three values as desired. The microprogram sequence from microword 8 through microword 15 executes a series of microinstructions that will determine the total number of ones in the three data fields, $V_0$, $V_1$, and $V_2$. These data values are loaded into Register 0, Register 1, and Register 2 of the Am2901 RAM. This microprogram uses Register 3 to hold the running partial summation as the number of ones in each word are counted. Register 4 in the Am2901 RAM is used as a working register to count the number of cycles in the algorithm. The contents of Register 0, Register 1, and Register 2 are not retained in the Am2901 memory; but they are destroyed during the execution of the algorithm. The flow diagram shown below is a summary of the algorithm as executed. The data value applied on the D input during the AND operation with each Register 0, 1, and 2 is used as a mask word.

As shown in the programming worksheet, if the program is executed it will finally reach memory word 15 and branch on itself thereafter. Memory word 15 is used to read register 3 such that the DATA DISPLAY can be used to read the total sum of ones in the $V_0$, $V_1$, and $V_2$ data fields. If field 6 of microprogram memory word 15 is changed from decimal 1 to decimal 2, instead of branching on itself at microprogram memory word 15, the microprogram will now continue from address 15 to address 0 and repeat the sequence. This allows the total sequence to be executed in the dynamic mode such that all the various instructions can be viewed using an oscilloscope. If the values $V_0$, $V_1$, and $V_2$ are to be changed, the kit should be switched from the dynamic mode to the static mode and these three data fields reloaded with the new values. Then, the kit can be switched back to the dynamic mode and the new sequence evaluated using an oscilloscope.

The oscilloscope can be synchronized to the total microprogram sequence by using the SYNC TEST turret terminal as a master sync point. This SYNC TEST point will provide one pulse each time the total sequence is executed. The ADDRESS SYNC turret terminal can be used as one trace on the oscilloscope to gain an instruction execution reference. For example, if the MEMORY ADDRESS switches are placed in the decimal 14 position, a sync pulse will be generated each time the algorithm jumps to the subroutine at microword 14 to increment Register 3. Likewise, if the MEMORY ADDRESS select switches are placed in the decimal 11 position, the end point of the four major cycles (decrement, register 4) can be referenced at the ADDRESS SYNC turret terminal.





## Am2900 KIT PROGRAMMING WORK SHEET

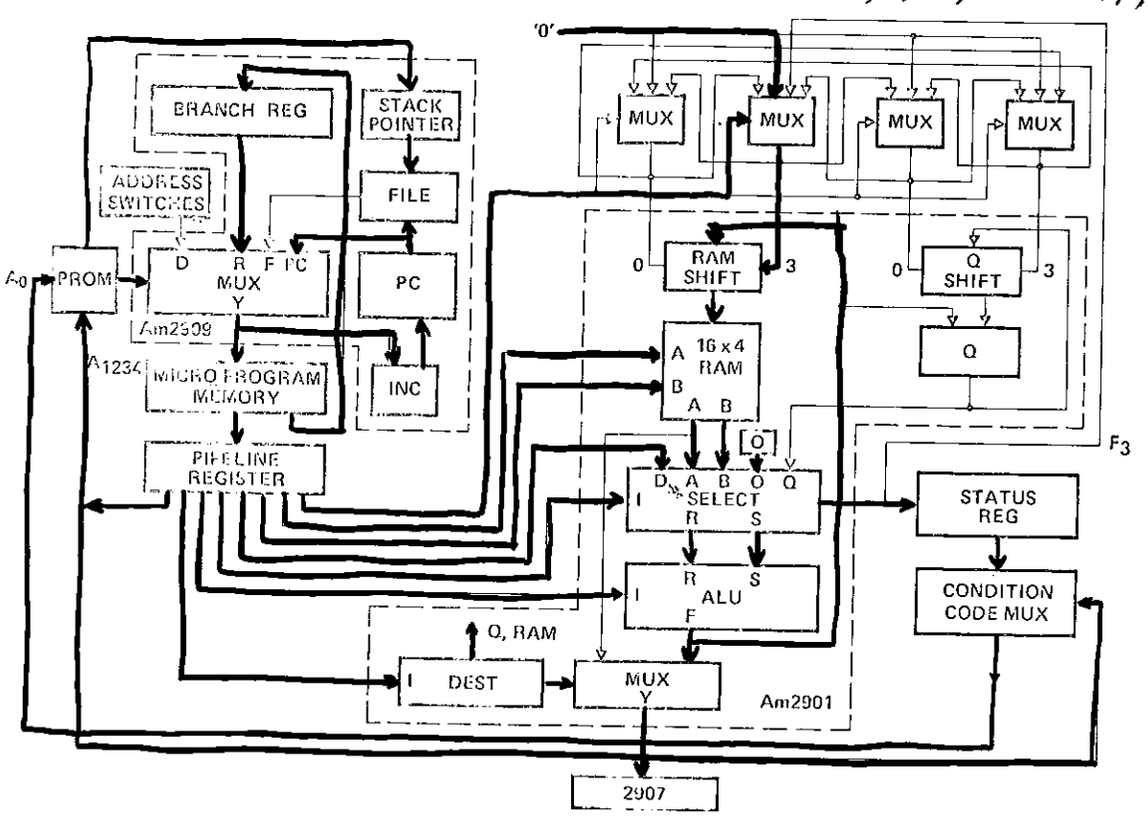
EXERCISE NUMBER: 10 NAME: NUMBER OF ONE'S IN $V_0, V_1$ & $V_2$

MICROPROGRAM MEMORY

| RAM & MUX | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | NOTES | |
|---|---|---|---|---|---|---|---|---|---|---|
| | BRANCH ADDRESS | NEXT INSTRUCTION CONTROL | DEST. SELECT (MUX1) | SOURCE SELECT (MUX0) | ALU ($C_n$) | 'A' | 'B' | 'D' | | |
| MEMORY ADDRESS | $BR_3$ $BR_2$ $BR_1$ $BR_0$ | $I_3$ $I_2$ $I_1$ $I_0$ | $I_8$ $I_7$ $I_6$ | $I_2$ $I_1$ $I_0$ | $I_5$ $I_4$ $I_3$ | $A_3$ $A_2$ $A_1$ $A_0$ | $B_3$ $B_2$ $B_1$ $B_0$ | $D_3$ $D_2$ $D_1$ $D_0$ | NEXT ADDRESS CONTROL | DATA CONTROL |
| 0 | | 0010 | 011 | 111 | 011 | | 0000 | $V_0$ | CONT | LD $V_0$ |
| 1 | | 0010 | 011 | 111 | 011 | | 0001 | $V_1$ | CONT | LD $V_1$ |
| 2 | | 0010 | 011 | 111 | 011 | | 0010 | $V_2$ | CONT | LD $V_2$ |
| 3 | | 0010 | 011 | 111 | 011 | | 0100 | 0100 | CONT | $R_4 = 4$ |
| 4 | | 0010 | 011 | 011 | 100 | | 0011 | | CONT | $R_3 = 0$ |
| 5 | | 0010 | 001 | 101 | 100 | 0000 | 0000 | 0001 | CONT | $R_0$ · MASK |
| 6 | 1110 | 0100 | 101 | 011 | 011 | | 0000 | | JSB14F/0 | $R_0 \leftarrow R_0/2$ |
| 7 | | 0010 | 001 | 101 | 100 | 0001 | 0001 | 0001 | CONT | $R_1$ · MASK |
| 8 | 1110 | 0100 | 101 | 011 | 011 | | 0001 | | JSB14F/0 | $R_1 \leftarrow R_1/2$ |
| 9 | | 0010 | 001 | 101 | 100 | 0010 | 0010 | 0001 | CONT | $R_2$ · MASK |
| 10 | 1110 | 0100 | 101 | 011 | 011 | | 0010 | | JSB14F/0 | $R_2 \leftarrow R_2/2$ |
| 11 | | 0010 | 011 | 011 | 001 | | 0100 | | CONT | DEC $R_4$ |
| 12 | 0101 | 0000 | 001 | | | | | | BR5F/0 | NO OP |
| 13 | 1111 | 0001 | 001 | | | | | | BR 15 | NO OP |
| 14 | | 0110 | 011 | 011 | 000 | | 0011 | | RTS | INC $R_3$ |
| 15 | 1111 | 0001 | 001 | 011 | 011 | | 0011 | | BR 15 | READ $R_3$ |

BLANK = DON'T CARE

INSTRUCTION JSB 14 IF F/0 (#6, 8, 10) @CONT (5, 7, 9)



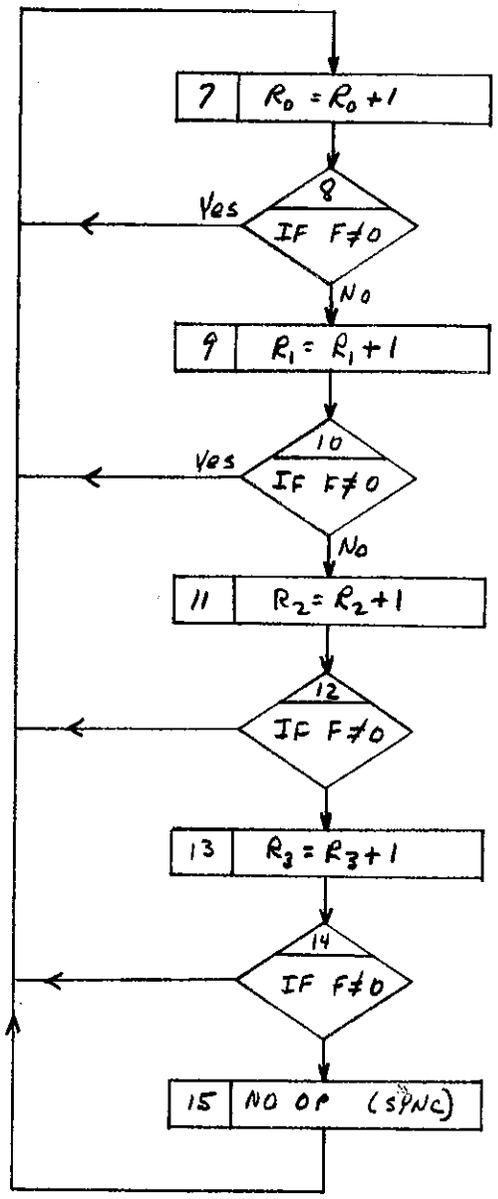



## PROGRAMMING EXERCISE #11 - 16-BIT PROGRAMMED COUNTER

This exercise demonstrates a technique for using four internal Am2901 RAM registers to emulate a 16-bit counter. In this example, Register 0 represents the least significant four bits. Then, Register 1 is used for the second four-bit field; Register 2 is used for the third four-bit field and Register 3 is used for the most significant four-bit field. The microprogramming sequence as demonstrated in this example could be used as a subroutine that is called each time an event occurs. A conditional Return-from-Subroutine could be used rather than conditional branch to word 7.

When the program reaches memory word 15, all four internal registers are at 0. At this point, $2^{16}$ calls of this subroutine would be required before microprogram state 15 will again be reached. The exercise is intended to be used in the dynamic mode; however, the user may wish to preload Register 0 through Register 3 with binary 15 (1111) so that the total branch path can be demonstrated in the static mode.

The flow diagram shown on the following page is a summary of the operation of this algorithm as programmed on the worksheet. By this point, the user should understand the technique required to initialize the sequence using the Am2900 Evaluation and Learning Kit.




EXERCISE # 11



## Am2900 KIT PROGRAMMING WORK SHEET

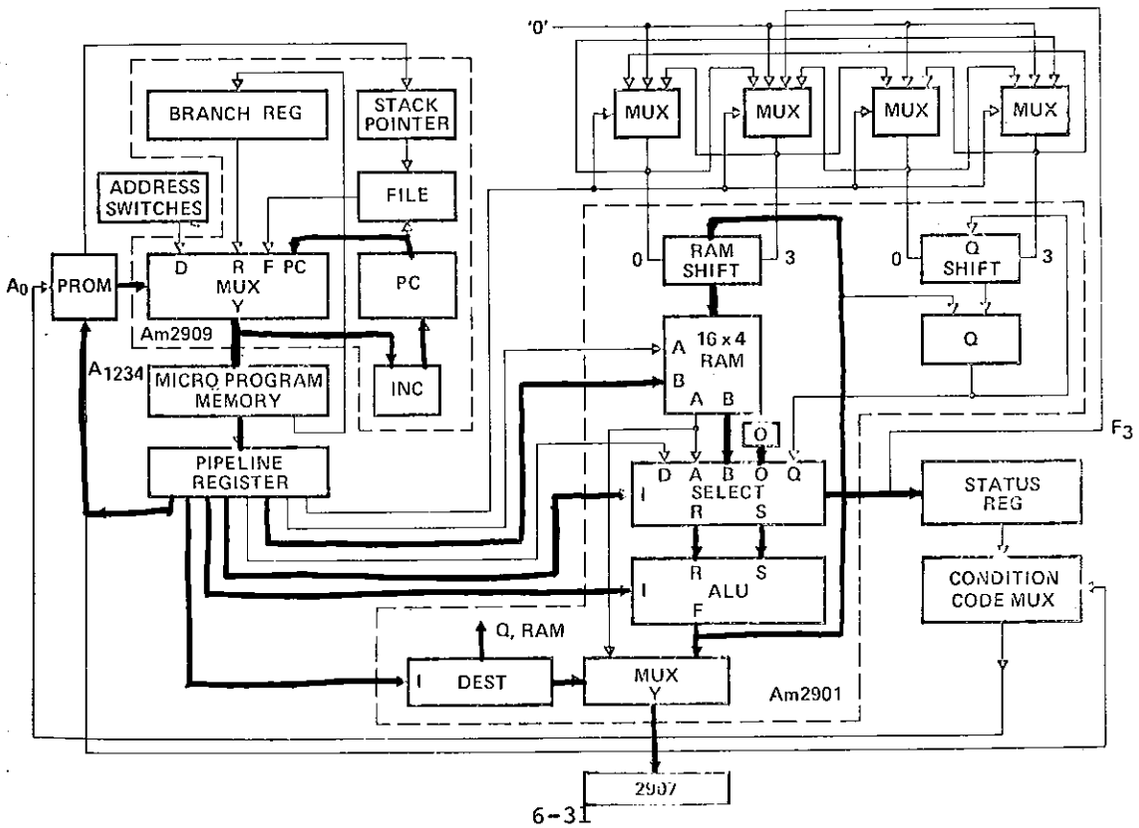
EXERCISE NUMBER: 11 NAME: 16-BIT COUNTER

### MICROPROGRAM MEMORY

| RAM & MUX / MEMORY ADDRESS | 7 BRANCH ADDRESS $BR_3$ | $BR_2$ | $BR_1$ | $BR_0$ | 6 NEXT μINSTRUCTION CONTROL $P_3$ | $P_2$ | $P_1$ | $P_0$ | 5 MUX1 | DEST. SELECT $I_8$ | $I_7$ | $I_6$ | 4 MUX0 | SOURCE SELECT $I_2$ | $I_1$ | $I_0$ | 3 $C_n$ | ALU $I_5$ | $I_4$ | $I_3$ | 2 'A' $A_3$ | $A_2$ | $A_1$ | $A_0$ | 1 'B' $B_3$ | $B_2$ | $B_1$ | $B_0$ | 0 'D' $D_3$ | $D_2$ | $D_1$ | $D_0$ | NOTES NEXT ADDRESS CONTROL | DATA CONTROL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | 0 | 0 | 1 | 0 | | 0 | 1 | 1 | | 0 | 1 | 1 | 1 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 | | | | | CONT | $R_0 = R_0 + 1$ |
| 8 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | BR7 F≠0 | |
| 9 | | | | | 0 | 0 | 1 | 0 | | 0 | 1 | 1 | | 0 | 1 | 1 | 1 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 1 | | | | | CONT | $R_1 = R_1 + 1$ |
| 10 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | BR7 F≠0 | |
| 11 | | | | | 0 | 0 | 1 | 0 | | 0 | 1 | 1 | | 0 | 1 | 1 | 1 | 0 | 0 | 0 | | | | | 0 | 0 | 1 | 0 | | | | | CONT | $R_2 = R_2 + 1$ |
| 12 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | BR7 F≠0 | |
| 13 | | | | | 0 | 0 | 1 | 0 | | 0 | 1 | 1 | | 0 | 1 | 1 | 1 | 0 | 0 | 0 | | | | | 0 | 0 | 1 | 1 | | | | | CONT | $R_3 = R_3 + 1$ |
| 14 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | BR7 F≠0 | |
| 15 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | BR7 | |

BLANK = DON'T CARE

INSTRUCTION $R_0 = R_0 + 1$



## Summary of Exercises

The exercises presented in Section VI of the Am2900 Evaluation and Learning Kit Instruction Manual have presented a number of different ideas associated with microprogramming. The user might attempt a number of other exercises to gain additional experience with this kit. Some ideas for these exercises are presented below.

1. Multiplication by a constant integer.

2. Division by a constant integer.

3. Counting the number of zeros in two register words.

4. Find the highest numeric value among four words.

5. Order three or four words in descending numerical order.

6. Perform a byte swap on one word.

7. Perform a logic compare on two words and count the number of bits not matching.

8. Add two registers and test for an arithmetic overflow

There are many such examples of small microprogram sequences of instructions that can be generated using this kit. Remember, however, the goal of this kit is to allow the engineer not familiar with microprogramming to grasp the concepts involved in microprogramming and not necessarily be able to use the kit to perform all possible combinations of microprogram sequences for instructions that can be suggested. Also, the Am2901, Am2907, and Am2909 dynamic performance can be evaluated.