

ED2900A

---

THE Am29203 EVALUATION BOARD  
EXERCISES & LABORATORIES



---

ED2900A

EVALUATION BOARD EXERCISES/LABORATORIES

- A. - Overview of Am29203 Evaluation Board
- B. - Laboratory 1      Introduction to Evaluation Board Monitor
- C. - Exercise 1        Am2910 Sequencer
- D. - Laboratory 2      Microprogramming the Sequencer
- E. - Exercise 2        Am29203 ALU
- F. - Laboratory 3      Microprogramming the ALU Basic Functions
  
- Appendix A            Evaluation Board Field Definitions



## ED2900A/B

OVERVIEW OF  
THE Am29203 EVALUATION BOARD  
FOR USE IN ED2900A/B LABORATORIES

- I The Am29203 Evaluation Board Characteristics
- II Architecture
- III The Primary System Architecture
- IV The Primary System Microword Format
- V The Primary System CCU Architecture
- VI Primary System Pipeline Registers
- VII Primary System Mapping PROM
  - A. Design Approach
  - B. Mapping PROM Layout
- VIII Primary System Writeable Control Store (WCS)
- IX The Primary System ALU Architecture
- X The Primary System Memory and I/O
- XI Microinstruction Field Overlay
- XII Microinstruction Field Encoding



### I. The Am29203 Evaluation Board Characteristics

- Stand-alone evaluation board for AMD bit-slice components
- Requires only power supply and CRT terminal
- Architecture represents typical 16-bit computer
- Built in Monitor allows:
  - Loading and displaying of writeable control store
  - Loading and displaying of macro memory
  - Loading and displaying of ALU registers
  - Loading and displaying of pipeline registers
  - Loading and displaying of the macro IR
  - Loading and displaying the Am2904 status registers
  - Setting breakpoints to control execution
  - Starting execution at any microaddress
  - Running built-in test routines
- Demonstrates microprogramming of the Am2900 family:
  - Am2910 sequencer
  - Am29203 arithmetic/logic unit (ALU)
  - Am2904 status and shift control unit

## II. Architecture

- See Figure EB-1
- Actually two systems
  - Two Am2910 sequencers
  - Two control stores and pipeline registers
  - One shared 16-bit Am29203-based ALU
- The Monitor controls the board operation
  - Transparent to the user
  - Allows interface to the controlling CRT
  - Executes the Monitor program (in microcode)
  - Controls execution of the Primary System
  - Allows examining and changing the Primary System state
- The Primary System is the system we will examine
  - Standard 16-bit architecture
  - Features Am29203, Am2910 and Am2904



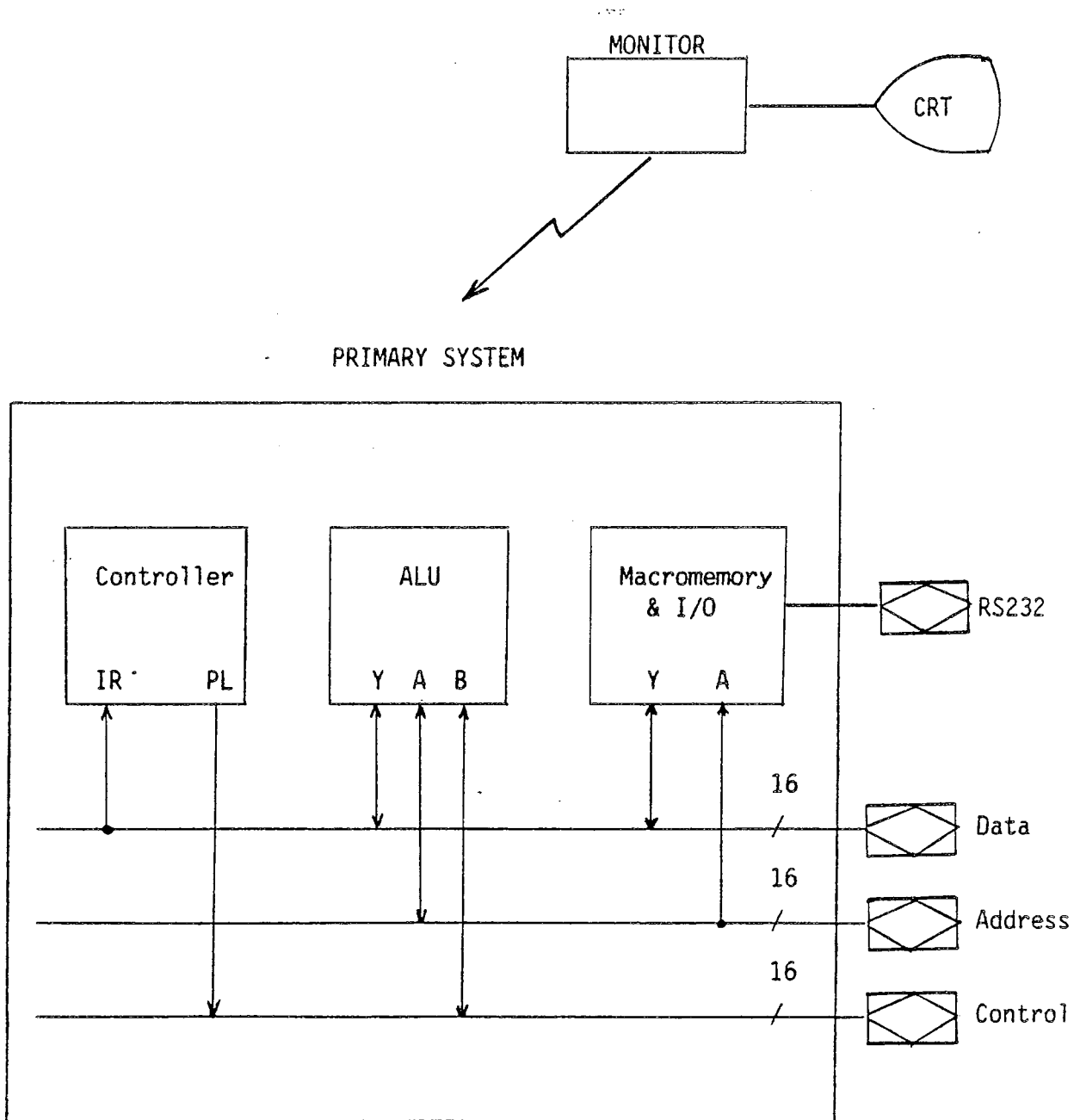


Figure EB-1. Evaluation Board Overview

---

### III. The Primary System Architecture

- See Figure EB-2
- CCU
  - Uses 8-bit opcode through mapping PROM
  - Uses Am2910 sequencer
  - Has 1K x 48 bits of writeable control store
  - Has separate pipeline register
  - Uses decoding on some pipeline fields
  - Condition codes come from the Am2904 test mux
  - Am2910 CCEN is controllable from pipeline for forced pass
- ALU
  - Am29203
  - Uses Am2904 for shift linkage and carry-in MUX
  - A & B addresses come from IR or pipeline
  - Addresses 1K on board RAM via A-bus
  - Data transfers to/from RAM via Y-bus
  - Receives constants from pipeline via B-bus
- I/O
  - Second I/O UART is connected to the Y-bus
  - Uses memory-addressed I/O
  - Can drive a CRT, printer or similar devices
- Miscellaneous Features
  - Many signals available at connectors for expansion
  - Macro instruction set can be downloaded into WCS

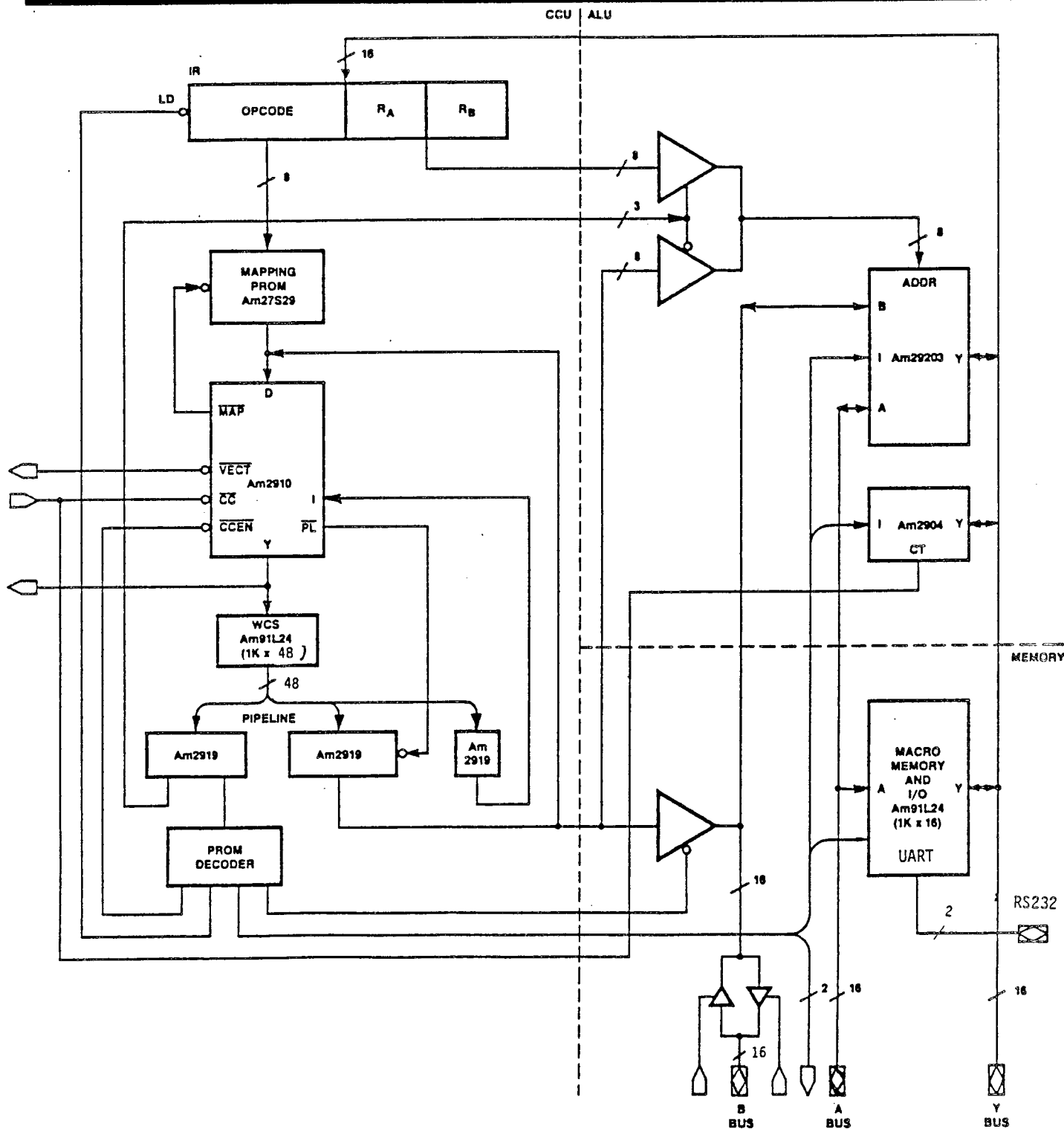


Figure EB-2. Primary-System Architecture

### IV. The Primary System Microword

- 48-bits wide
- Bits 31-16 control the Am2904
  - Bits 19-16 are three overlaid fields
  - Bit 21 selects command field
  - Command field decoded by PROM
- Bit 15 is breakpoint bit (set and cleared by Monitor)
- Bits 13-4 make up the branch address field
  - Can be turned off by  $\overline{PL}$  on Am2910
  - Bits 11-4 are three overlaid fields
    - Lowest 8 bits of branch address
    - ALU register addresses
    - Constant value for ALU

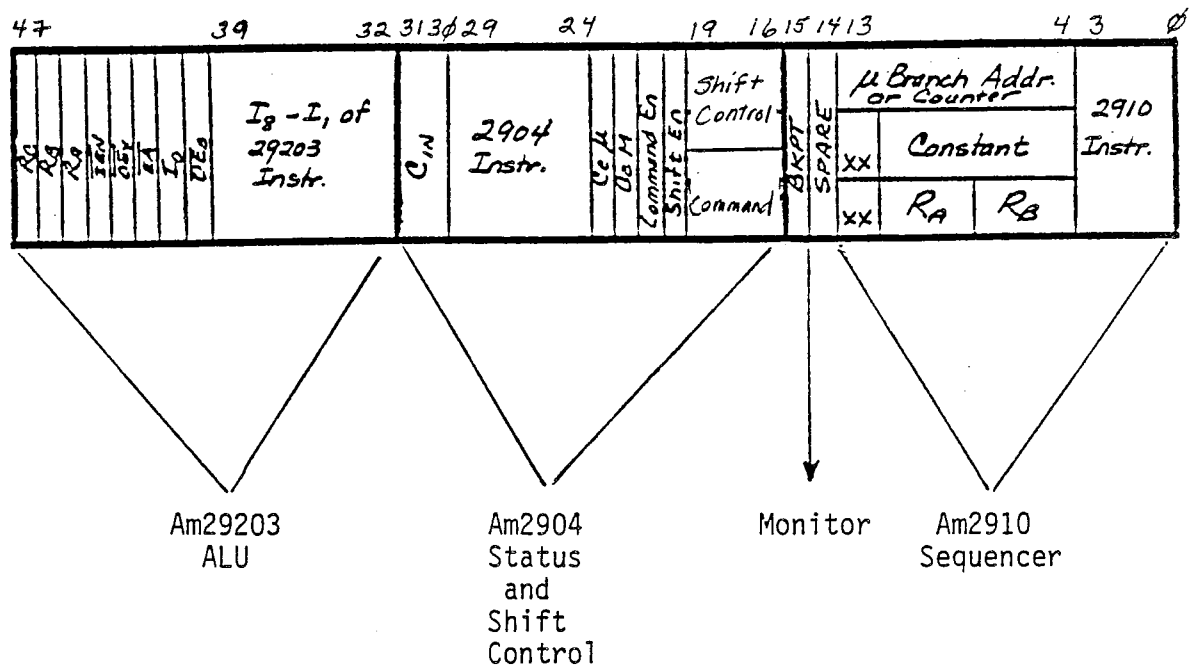


Figure EB-3.

### V. The Primary System CCU Architecture

- See Figure EB-4
- 8-bits of machine level opcode
- 10-bits of control-store addressing
- Two Am2910 D-input sources controlled by Am2910  $\overline{OE}$  lines
  - Routine starting address from mapping PROM
  - Branch address from pipeline register
- All Am2910 instructions available
- Vector-map-enable not used on this board
- Condition-code input comes from Am2904
- $\overline{CCEN}$  controlled from pipeline for forced pass
- 1K x 48-bit Writeable Control Store (WCS)
- Separate pipeline register has tri-state section
- A PROM is used to decode a command field of the microinstruction

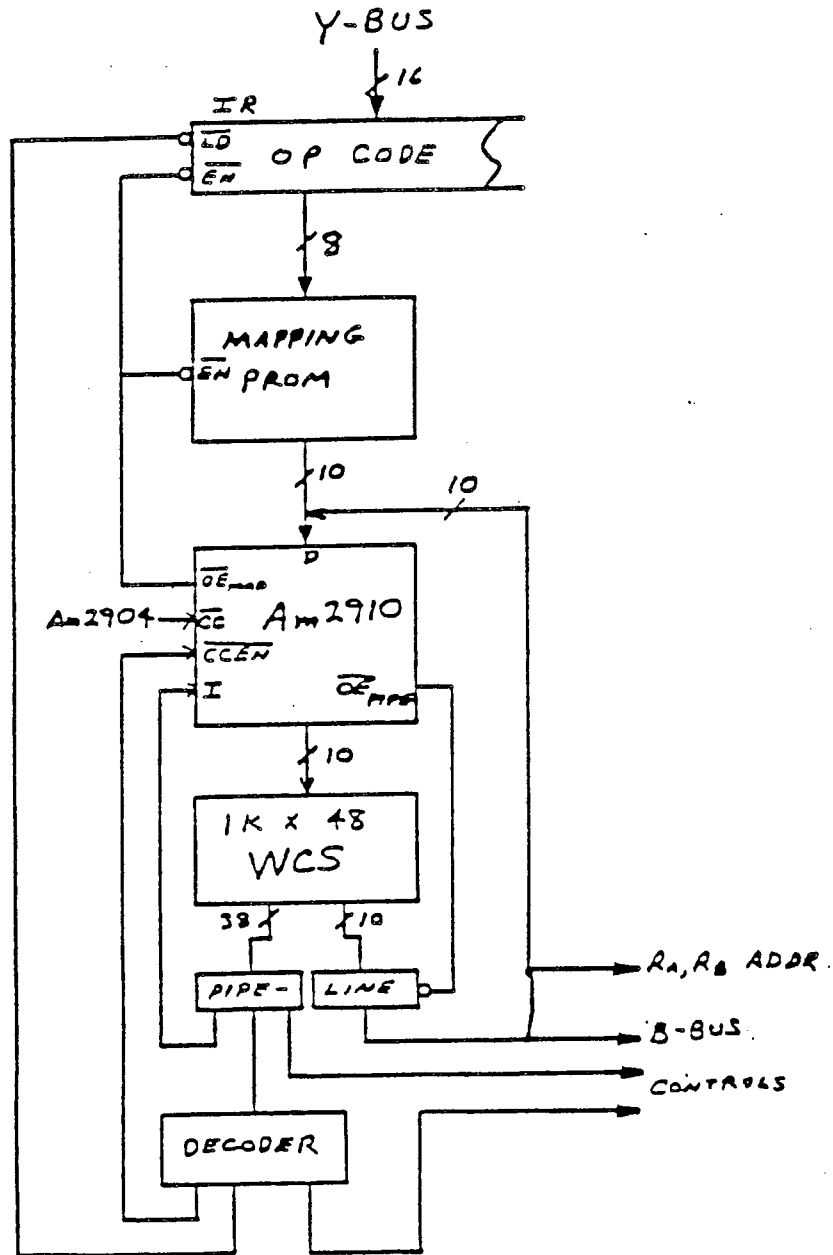


Figure EB-4. Primary-System CCU Architecture

### Primary System Sequencer

- Am2910 sequencer is used. See Figure EB-5.
- 10 of the 12 address lines are used
- $\overline{RLD}$  tied high (unused)
- $\overline{CC}$  driven from Am2904 (condition code MUX and test)
- $\overline{CCEN}$  driven from pipeline through decoding PROM
- Next-address instructions come from pipeline
- $\overline{OE}$  controlled by Monitor
- $\overline{PL}$  controls tri-state section of pipeline
- $\overline{MAP}$  controls output of mapping PROM
- $\overline{VECT}$  not used
- CI (carry-in) tied high (normal operation)
- $\overline{FULL}$  not used

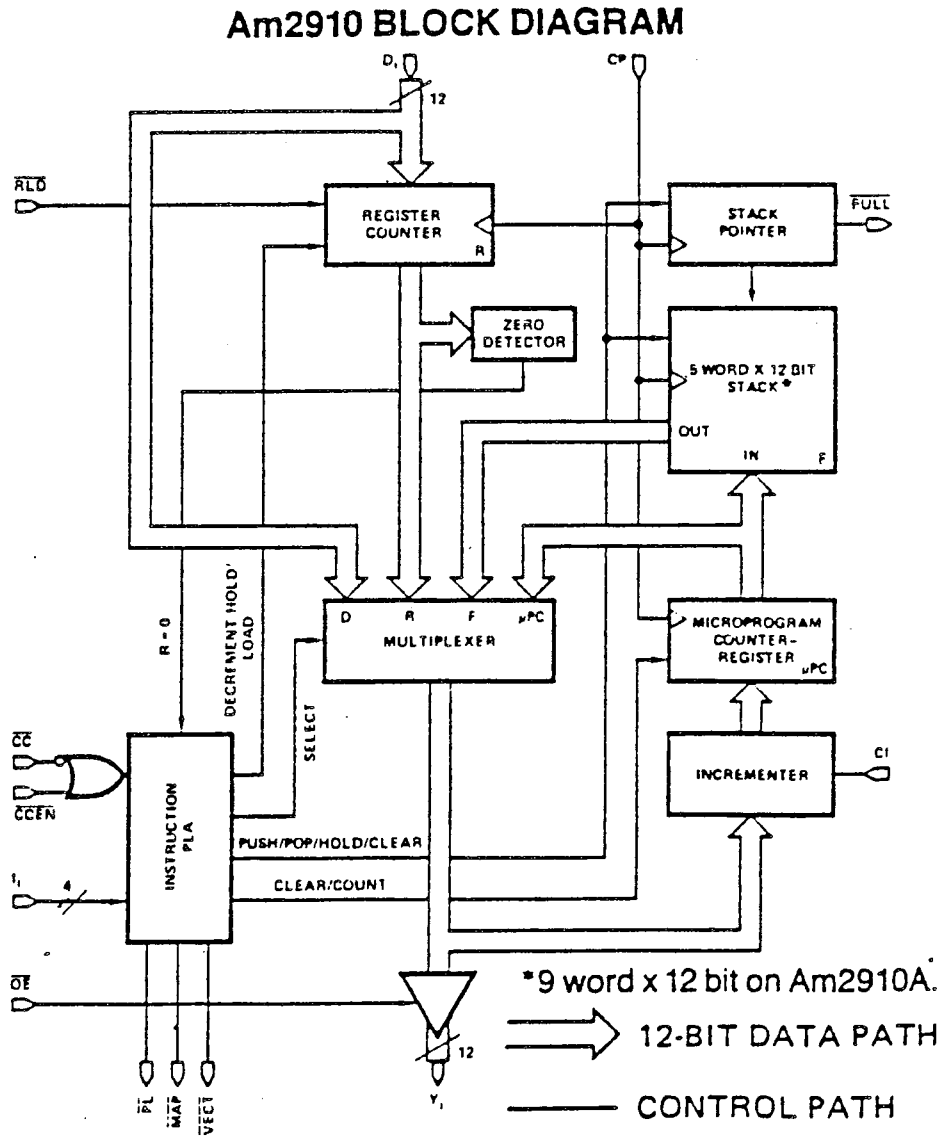


Figure EB-5. Am2910 Structure



### Primary System Sequencer - Cont'd

- All 16 Am2910 instructions are usable with some restrictions. See Figure EB-6.
- CJV will conditionally jump to address 3FF
- $\overline{\text{CCEN}}$  must be used for forced pass
- No forced-fail condition is available (i.e. you cannot do a PUSH without loading the counter)
- Counter field is only 10-bits wide (max count = 1023)

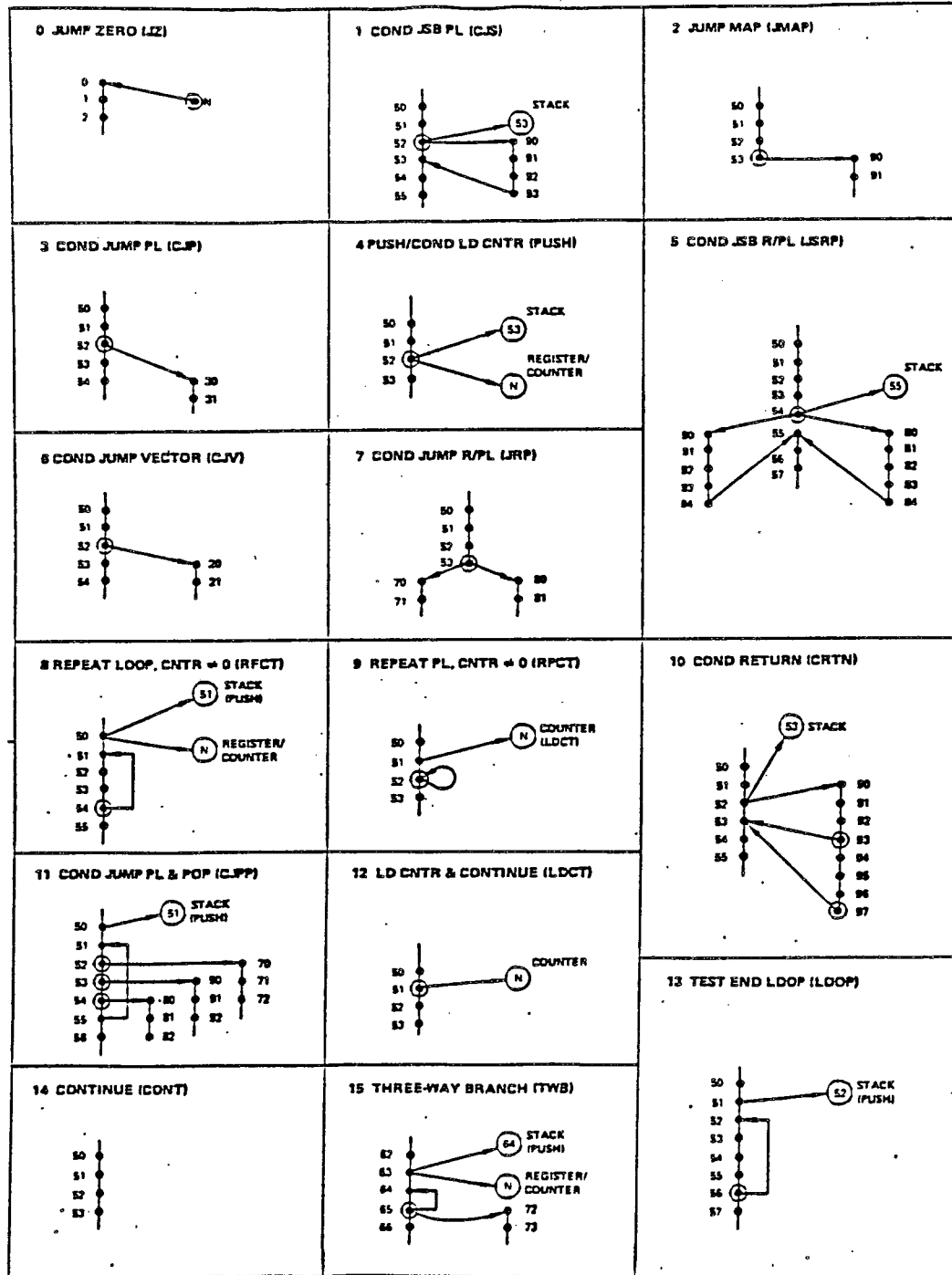


Figure EB-6. Am2910 Instruction Set

## VI. Primary System Pipeline Registers

- Only the Branch Address/Count field is tri-stated:
  - Tri-state enable comes from Am2910  $\overline{PL}$
- Am2910 instruction field always enabled
- $\overline{CCEN}$  comes from decoding PROM
- $\overline{LDIR}$  (load the instruction register) comes from decoding PROM
- We will discuss the microword details shortly

## VII. Primary System Mapping PROM

(see Figure EB-7)

### A. Design Approach

- Maps opcodes to WCS addresses
- Based on a compromise 8-bit PROM approach
  - Uses only one chip
  - Maps to every words (even addresses) in WCS
- 8-bit opcode mapped to 10-bit microcode address -- achieved by two constraints:
  - All output addresses are even. The LSB of the address is tied low.
  - MSB of opcode is tied to MSB of output address so that  
128 opcodes with MSB=0 map to any of 256 even addresses <512  
& 128 opcodes with MSB=1 map to any of 256 even addresses >=512

**B. Mapping PROM Layout**

- Upper 512 words of WCS are loaded with example microcode
  - All op codes start with 1
  - Automatically loaded on reset
  - Manually loaded using LI
  - Supports example macro instruction set
  - Op codes mapped to fit microroutines
  
- Lower 512 words intended for user routines
  - All op codes start with 0
  - All 512 locations available
  - Op codes mapped to evenly spaced addresses
  - Four microwords are available for each op code
  - Larger routines invalidate the next op code(s)
  
- The user can replace the PROM to provide other mappings

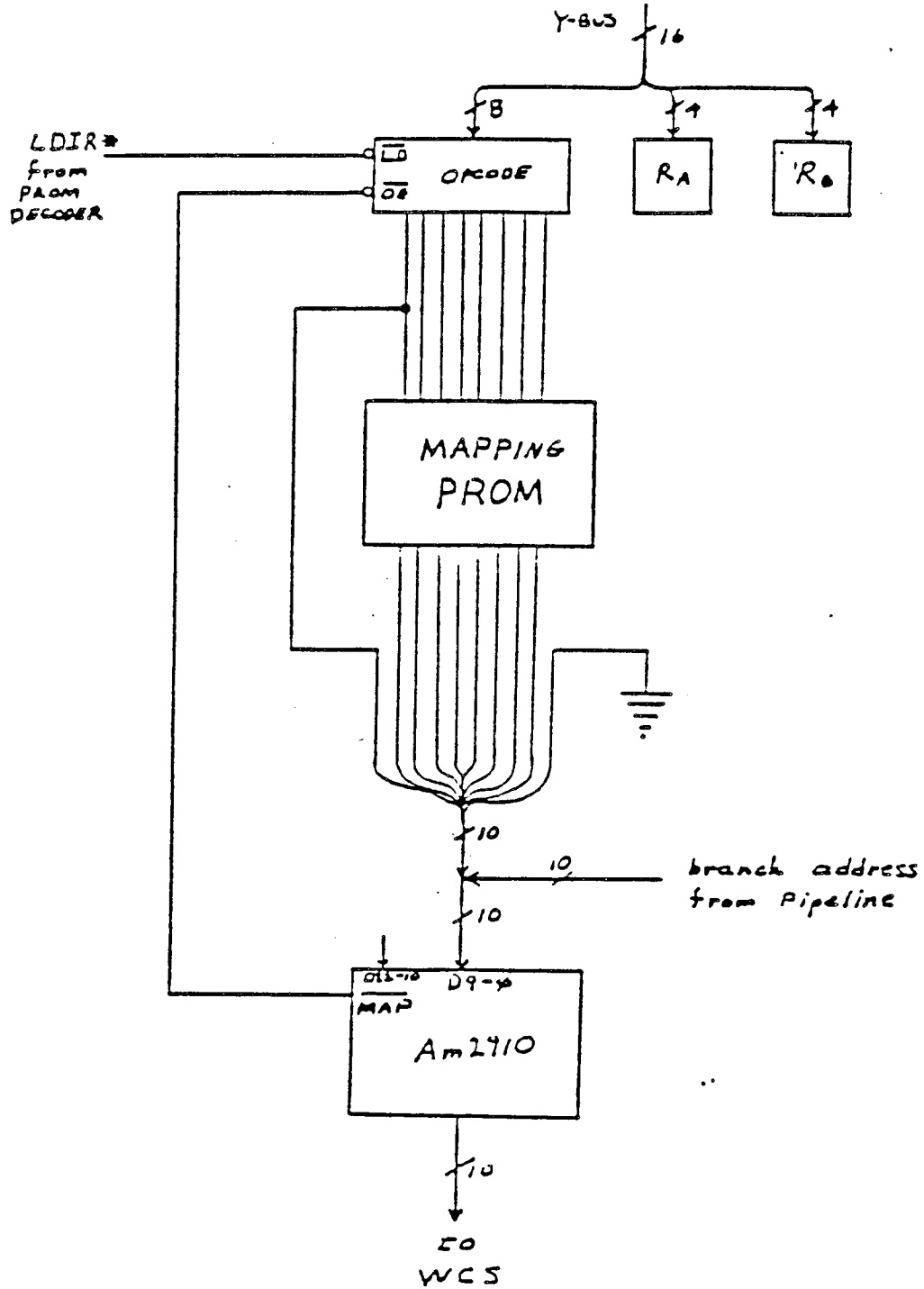


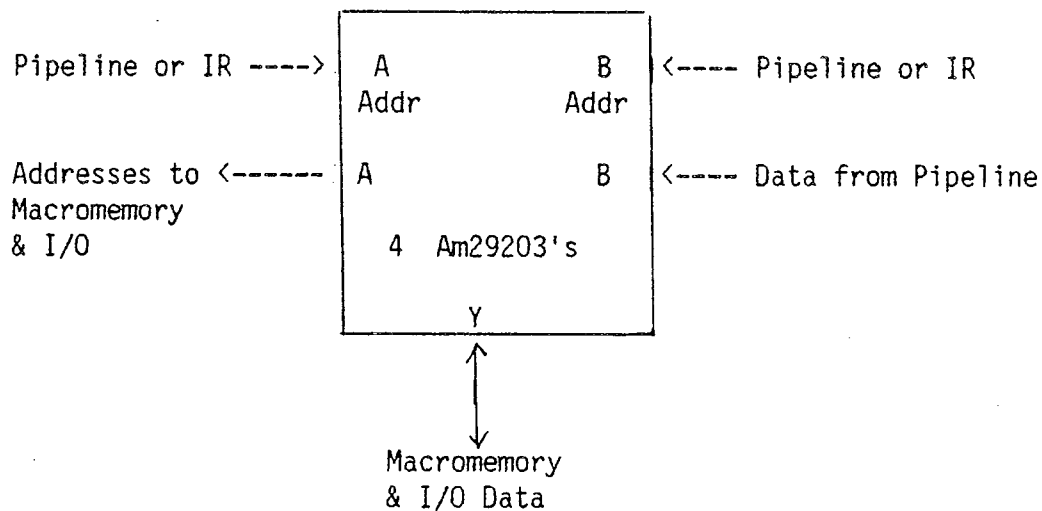
Figure EB-7. Mapping PROM Configuration

### VIII. Primary System Writeable Control Store (WCS)

- 1K x 48-bit RAM used for storing microcode
- Loaded under Monitor control
- Upper 512 words loaded with example microcode
  - Loaded at board initialization
  - Can be overwritten by user routines
- In production environment, usually ROMs, PROMs, or Registered PROMs are used

### IX. The Primary System ALU Architecture

- Four Am29203s connected in ripple-carry architecture. See Figure EB-8.
- Y-bus is the primary data transfer bus
- A-bus is used for addressing memory and I/O
  - No explicit MAR is used
- B-bus is a data input from the pipeline
- A, B addresses can come from the IR or pipeline
- Am2904 provides all ALU support
  - Carry-in MUX
  - RAM and Q shifter MUXs
  - Micro and Macro status registers
  - Condition-code MUX and test selection
- Y-bus allows Am2904 contents to be saved in memory





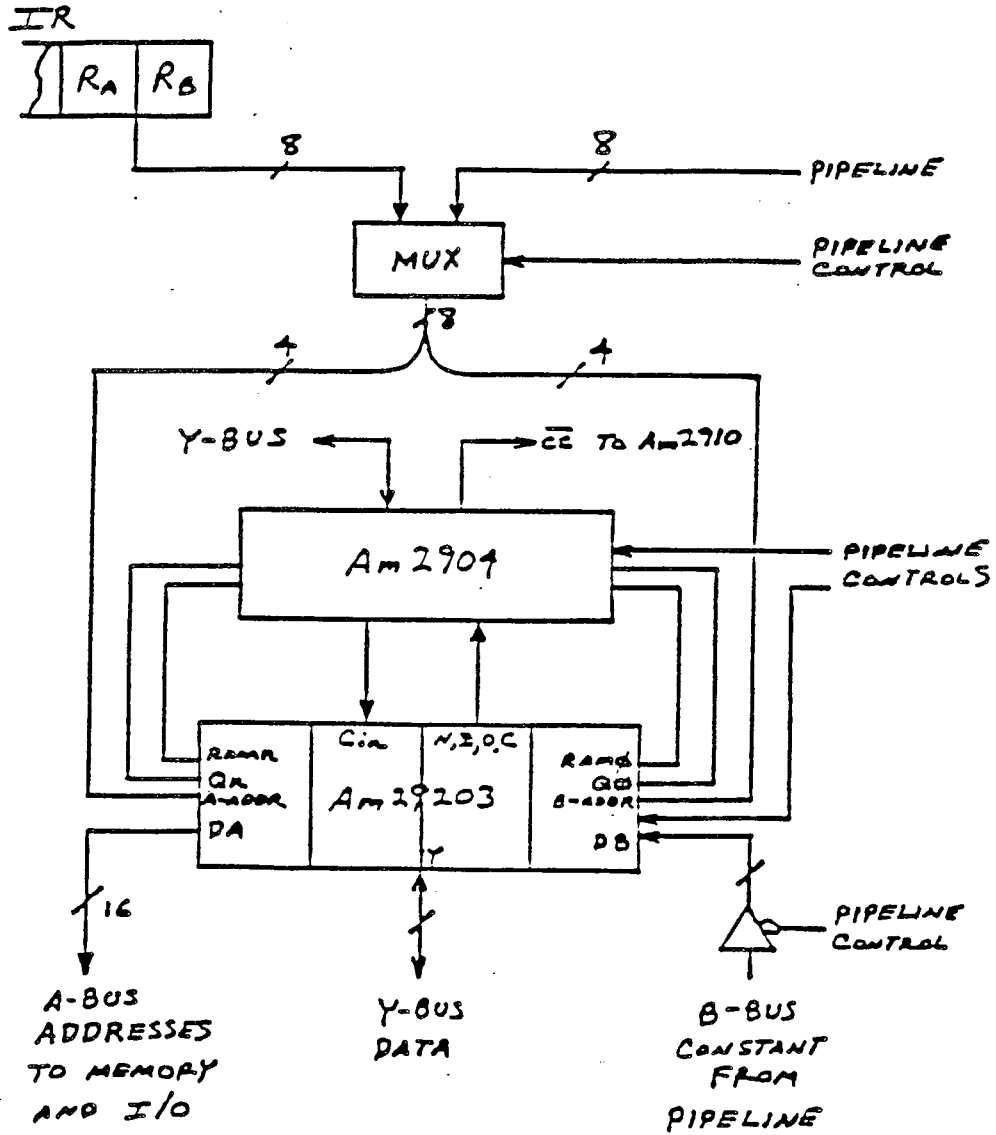


Figure EB-8. Primary-System ALU Architecture

### X. The Primary System Memory and I/O

- All addressing via the A-bus
- I/O addressed just like memory
- Memory enable and read/write control from decoding PROM
- Address space divided
  - 1K x 16-bit RAM
  - 4K x 16-bit PROM for microcode examples  
(downloaded to WCS on Monitor command)
  - Two I/O addresses for second UART
  - 32K for offboard memory
- All data transfers via the Y-bus

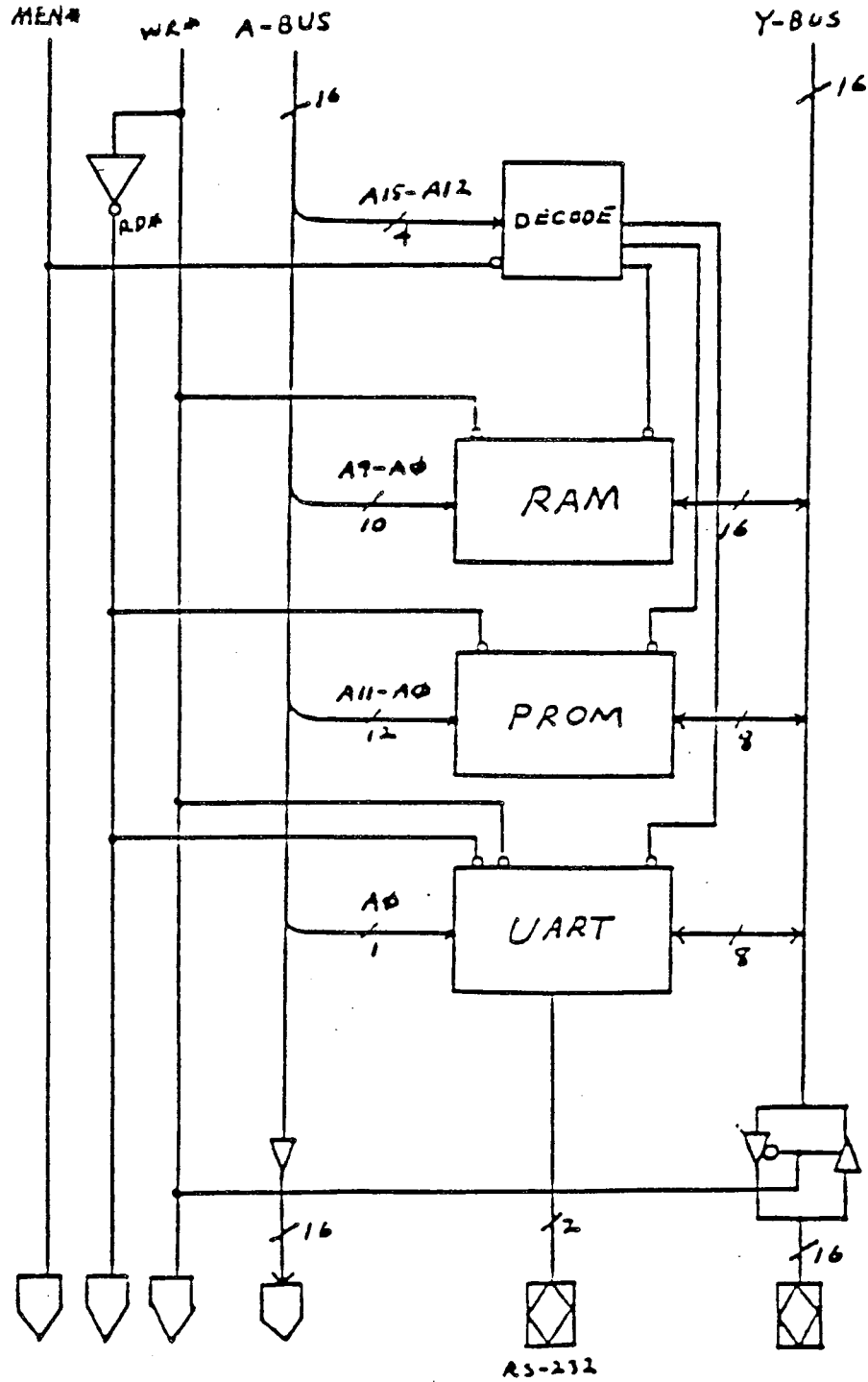


Figure EB-9. Memory and I/O Architecture

### XI. Microinstruction Field Overlays

- Two areas of overlay occur on the Evaluation Board
- Bits 13-4 actually have four overlaid fields!
  - Micro Branch Address for Am2910
  - Counter value (10-bits only) for Am2910
  - Ra and Rb register addresses for Am29203
  - Constant value for the Am29203 via the B-bus
  - For example, CJP cannot be done if Ra, Rb are specified
- These fields require bit steering:
  - Branch Address selected by Am2910 instruction
  - Counter selected by Am2910 instruction
  - Ra, Rb selected by bits 47-45
  - Constant is selected by  $\overline{CON}$  from the decoder
- Bits 19-16 have two overlaid fields
  - Am2904 Shift Control
  - Encoded-command field
- Status-Enable field selected by bit 22
- Shift-Control field selected by bit 20
- Encoded-Command field selected by bit 21
- This overlaying imposes some limitations on parallel operations

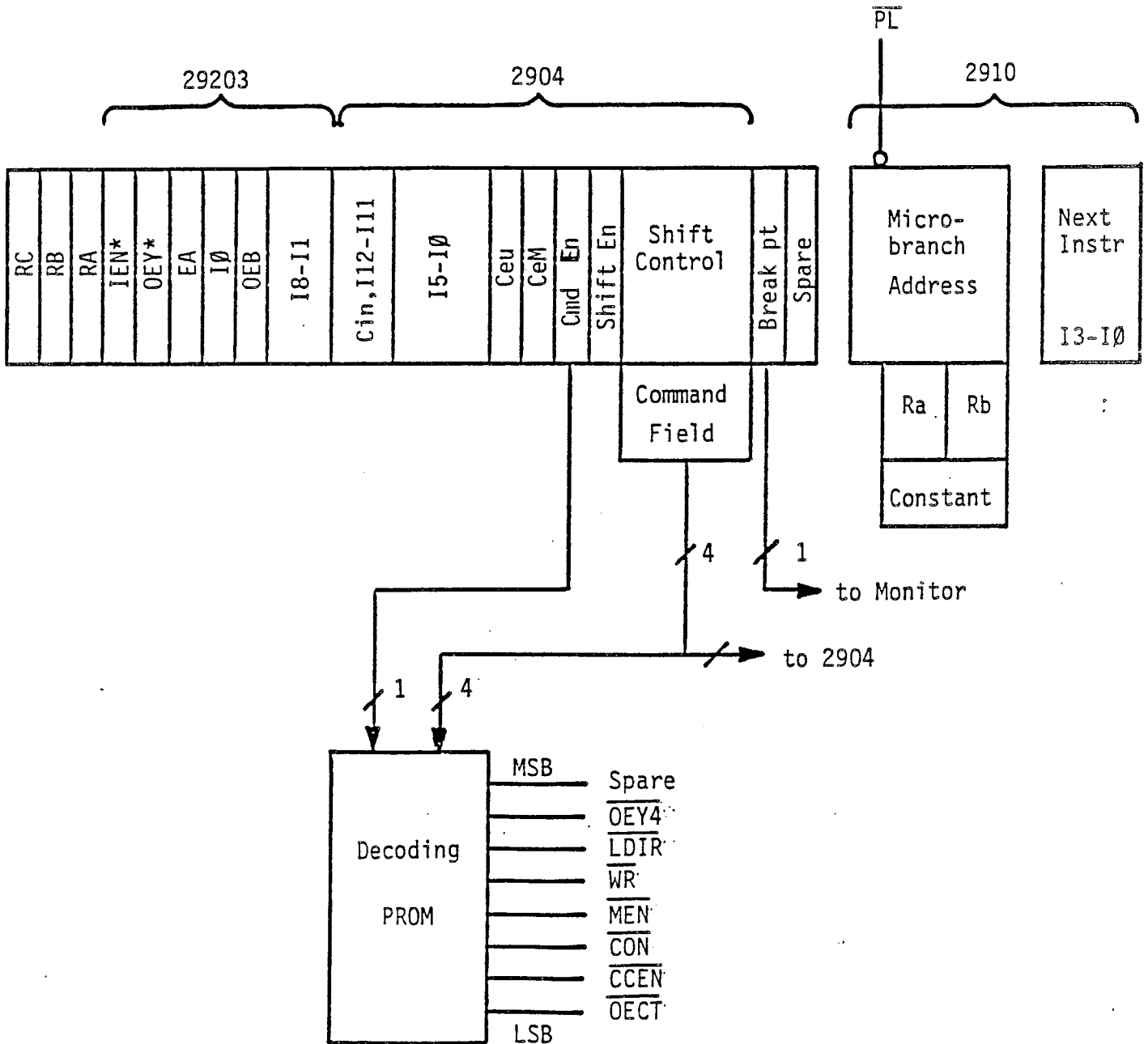


Figure EB-10. Microword and Decoding PROM

## XII. Microinstruction Field Encoding

- Encoding is heavily used on the Command field, bits 19-16
- Microword is effectively reduced by 3 bits:
  - Only seven control bits from the decoder are used
  - Four bits are needed to generate these seven
  - Steering bit (21) would be needed anyway
  - Adding 3 bits to the pipeline would have added two more ICs: memory plus pipeline register
- The seven resources controlled are:
  - $\overline{OEY4}$  = Am2904 Status Output Enable
  - $\overline{LDIR}$  = Load Macroinstruction Register
  - $\overline{WR}$  = Memory write/read
  - $\overline{MEN}$  = Memory enable
  - $\overline{CON}$  = Enable constant field to Am29203
  - $\overline{CCEN}$  = Forced pass to the Am2910
  - $\overline{OECT}$  = Am2904 condition-code-test mux enable
- Combinations of these seven are decoded to create fourteen meaningful commands

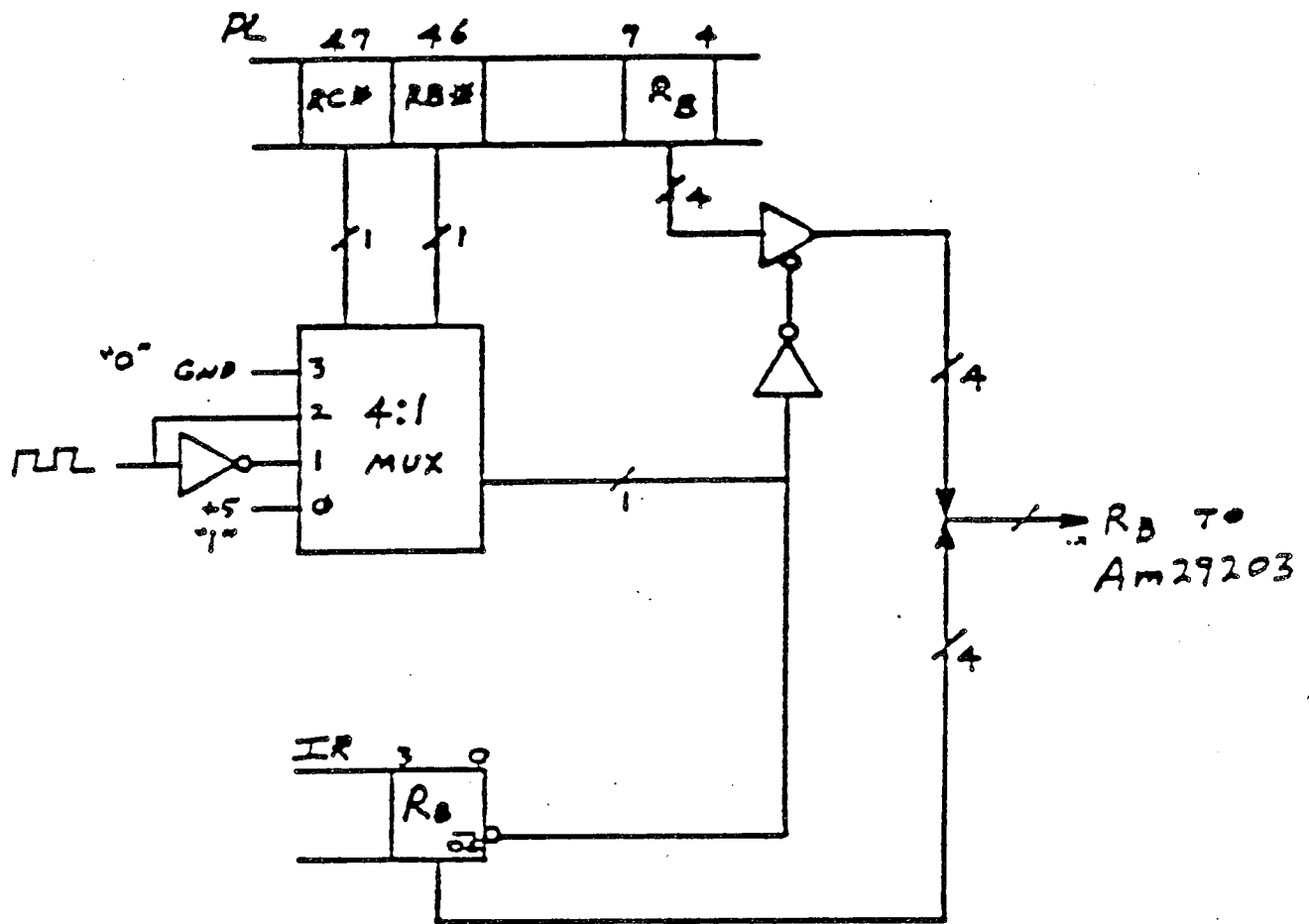
### More Microinstruction Field Encoding

- Bits 47-46 control Rb and Rc for three-address instructions
- These bits select the Am29203 Rb-address from the IR or from the pipeline
- These bits are encoded to provide four possible conditions
  - 00 => Rb comes from the pipeline (2-address)
  - 01 => Rb comes from IR first half cycle,  
Rb comes from pipeline second half cycle
  - 10 => Rb comes from pipeline first half cycle,  
Rb comes from IR second half cycle
  - 11 => Rb comes from the IR (2-address)

Table EB-1. Decoding-PROM Map

Addr	S p a r e	O Y 4 *	L I R *	W *	M *	C N *	C T *	O E N T *	.DEF	Hex Value	Explanation
00	1	0	1	1	1	1	1	1	OEY04	BF	Enable 2904 Y-output
01	1	1	0	1	1	1	1	1	LDIR	DF	Load Instruction Register (IR)
02	1	1	0	1	1	0	1	1	CONAB	DB	Register Address thru ALU to IR
03	1	1	1	1	0	1	1	1	RDMEM	F7	Read Memory
04	1	1	1	0	0	1	1	1	WRTMEM	E7	Write to memory
05	1	1	1	1	1	0	1	1	CONBUS	FB	Enable constant to B-bus
06	1	1	0	1	0	1	1	1	IFTCH	D7	Instruction fetch
07	0	1	1	1	1	1	1	1	SPARE	7F	Enable spare command line
08	1	1	1	1	1	1	0	1	SCCEN	FD	CCEN input to Am2910
09	1	1	1	1	1	1	0	0	ALUTST	FC	Enable 2904 CT to 2910 CC input
0A	1	1	1	1	1	1	1	1	READ	FF	Read enable
0B	1	1	1	0	1	1	1	1	WRITE	EF	Write enable
0C	1	0	1	0	0	1	1	1	SAVSTAT	A7	Write 2904 status to memory
0D	1	1	1	0	0	0	1	1	SAVECON	E3	Write constant to memory
0E	1	1	1	1	1	1	1	1		FF	Not used
0F	1	1	1	1	1	1	1	1		FF	Not used
10	1	1	1	1	1	1	1	1		FF	Not enabled
11	1	1	1	1	1	1	1	1		FF	Not enabled
.											
.											
1E	1	1	1	1	1	1	1	1		FF	Not enabled
1F	1	1	1	1	1	1	1	1		FF	Not enabled





$R_{C\#}$	$R_{B\#}$	$R_B$
0	0	PL
0	1	IR PL
1	0	PL IR
1	1	IR

} 3-ADDRESS

Figure EB-11. Register Address Source Encoding

**ED2900A****B. - Laboratory 1 - Introduction to Evaluation Board Monitor**

- Make certain power is connected to the board.
- Make certain that the CRT is connected.
- Turn on the power. Allow the CRT to warm up.
- Press the reset button on the board.
- You should see the prompt ">" followed by a short summary of the available commands
- If this message doesn't appear, get help.

---

### Monitor Command Summary

- The Evaluation Board Monitor prompt is a ">".
- Commands are terminated by a carriage return <CR>.
- All displays are in hexadecimal (Base 16).
- Four types of error control/recovery are available:
  - The ESC key aborts any command.
  - The backspace key can be used to correct input.
  - Keep typing. Only the last 4 hex digits are used.
  - Illegal commands are ignored and "beeped".
- The major commands (entered after the >) are:
  - >L - load
  - >D - display
  - >G - go - execute microcode
  - >T - test - run test routines
  - >Z - zeros all registers.
- Except for "T", these commands need further input.
- Evaluation Board resources are identified by:
  - R - Registers
  - M - Main memory (not control store)
  - C - Control store
  - P - Pipeline
  - B - Breakpoint
  - I - Instruction set (macro)
  - A - Address of current pipeline value
  - N - Address of Next pipeline value

### Using the Display Register Command

- Type DR on the terminal. The current register values are displayed in the following format:

```
>DR REG:
 0   1   2   3   4   5   6   .... D   E   F
0000 0000 0000 0000 0000 0000 0000      0000 0000 0000
  Q   IR   MS   US(VCNZ)
0000 0000   0   0
```

The first row displays the ALU register numbers.

The second row displays the ALU register contents.

The third and fourth rows show the contents of

Q - The Am29203 Q-register

IR - The Macro Instruction Register

MS - The Am2904 Macro Status Register (4-bits)

US - The Am2904 Micro Status Register (4-bits)

The sequence, "VCNZ", reminds you of the bit-sequence of the status bits in the MS and US registers:

overflow, carry, sign, zero

### Using the Display Main Memory Command

- Type DM on the terminal.
- You are prompted for a starting address

>DM ADDR:

- Enter up to 4 hex digits followed by a carriage return.
- The starting address and the contents of the next eight sequential locations are displayed.
- Typing any key displays the next 8 locations.
- Typing the ESC key terminates this mode.
- Display the 24 locations starting with address 200.

### Using the Display Control Store Command

- Operation is the same as for main memory.
- The display format is slightly different.
- Type DC on the terminal.
- You are prompted for a starting address

>DC ADDR:

- Enter up to 4 hex digits followed by a carriage return.
- The address and one 48-bit WCS word are displayed.
- Typing the ESC key terminates this mode.
- Display five locations starting with address 100.

**Using the Display Pipeline Command**

- Type DP on the terminal.
- The 48-bit pipeline is displayed.
- This is the actual contents of the pipeline register during the current microcycle (that is about to be executed).

**Using the Display Address Command**

- Type DA on the terminal.
- The address of the instruction in the pipeline is displayed.

**Using the Display Next Address Command**

- Type DN on the terminal.
- The address of the next instruction to be put into the pipeline is displayed.

**Using the Display Breakpoints Command**

- Type DB on the terminal.
- The addresses of all microinstructions with the breakpoint set are displayed.

### Using the Load Main Memory Command

- Type LM on the terminal.
- You are prompted for an address:

```
>LM ADDR:
```

- Enter the address, terminated by a <CR> and you are prompted for data:

```
>LM ADDR: 037<CR>  
DATA:
```

- Enter up to 4 hex digits terminated by a <CR>.

```
>LM ADDR: 037<CR>  
DATA: FFFF<CR>  
DATA:
```

- Data for the next sequential location is prompted for. Use ESC to quit.
- Now load the numbers 0 through F into the sixteen memory locations beginning at 200.
- Use DM to verify your actions.

### Using the Load Pipeline Command

- Type LP on the terminal.
- You are prompted for data:

```
>LP DATA:
```

- Enter up to 12 hex digits in groups of 4, separated by <SPACE> or by <CR> (which is not echoed):

```
>LP DATA: FFFF<CR> 1234<CR> ABCD<CR>  
>
```

or

```
>LP DATA: FFFF 1234 ABCD<CR>  
>
```

- Now load the following word into the pipeline register:

```
AAAA BBBB CCCC
```

- Use DP to verify your actions.

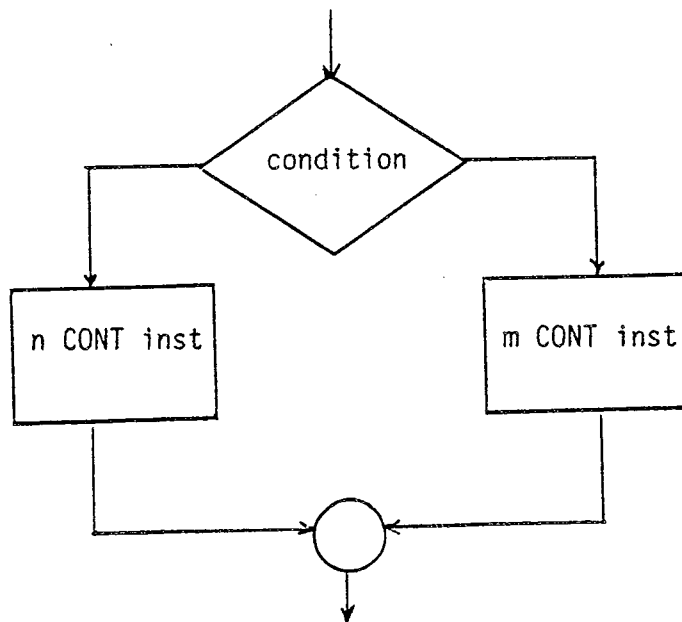
### Using the Load Instructions Command

- Type LI on the terminal.
- The example microcode for a macro instruction set is loaded from PROM into WCS from 0000 to 0200.
- Now verify the load using DC.



## Exercise 1 (cont'd)

3. Write a partial microroutine for an IF - THEN - ELSE Structure; i.e., if a condition is true, then execute n CONT instructions. If the condition is false then execute m CONT instructions:



### Sequencer Microword Fields

- Only the sequencer fields are programmed
  - Bits 3-0 = Am2910 instruction field
  
  - Bits 13-4 = Branch address & counter field
  
  - Bits 23-20 = Command enable field and status latch controls
    - Use "C" to allow conditional tests
    - Use "E" for forced pass via  $\overline{CCEN}$
  
- The Am2903 macro status register is used to provide true and false test conditions.
  
- Bits 23-22 must be "1" to prevent changing the Am2904 status registers inadvertently.
  
- Bit 15 (breakpoint) must be loaded with "1".
  
- All other fields are don't cares, set to "1".
  
- It is suggested that the Standard Evaluation Board Microinstruction Format be used for clarity

### Specific Laboratory 2 Exercises

1. Starting at micromemory address 0, execute 4 continue instructions.
  
2. Add a fifth instruction to the above microprogram to branch back to address 0 unconditionally.
  
3. Starting at micromemory address 10, execute a microprogram which loops on one microinstruction 5 times
  - a) using RPCT
  - b) using RFCT

Don't forget to set the counter before you begin.

4. Generate a 3-instruction loop using the LOOP instruction.

- The monitor enters Trace mode. The first microinstruction is fetched into the pipeline to be executed. The address, pipeline, and registers are displayed. Any key causes the next microstep to occur. The ESC key terminates the process.
  
- Single step through the routine, watching the address and pipeline register contents.
  
- For conditional statements, exercise both options by changing the contents of the "S" register using LR.

### F. - Laboratory 3 - Microprogramming the ALU Basic Functions

- The purpose of this laboratory is to provide an understanding of the Am29203 ALU and associated operations through the use of microprogramming. The laboratory consists of 6 exercises starting with a set of simple operations and ending with special Am29203 operations.
  
- 1. Using individually selected initial values entered via the monitor: write, run, and debug each register transfer language statement for the specified microcode. Define each operation with comments.

Initially, load the general purpose, IR, S, U and Q registers with values and check these values by means of a DR command.

## Worksheet for Exercise 2

<u>Microprogram Address</u>	<u>Microcode</u>
n	<u>7</u> <u>F</u> <u>3</u> <u>F</u>
n+1	<u>3</u> <u>F</u> <u>F</u> <u>F</u>
n+2	
n+3	
n+4	<u>1</u> <u>F</u> <u>1</u> <u>2</u>
n+5	
n+6	
n+7	
n+8	
n+9	
n+A	
n+B	
n+C	
n+D	
n+E	

**Worksheet for Exercise 3**

n	_____	_____	_____
n+1	_____	_____	_____
n+2	_____	_____	_____
n+3	_____	_____	_____
n	_____	_____	_____
n+1	_____	_____	_____
n+2	_____	_____	_____
n+3	_____	_____	_____

**Worksheet for Exercise 4**

n	_____	_____	_____
n+1	_____	_____	_____
n+2	_____	_____	_____

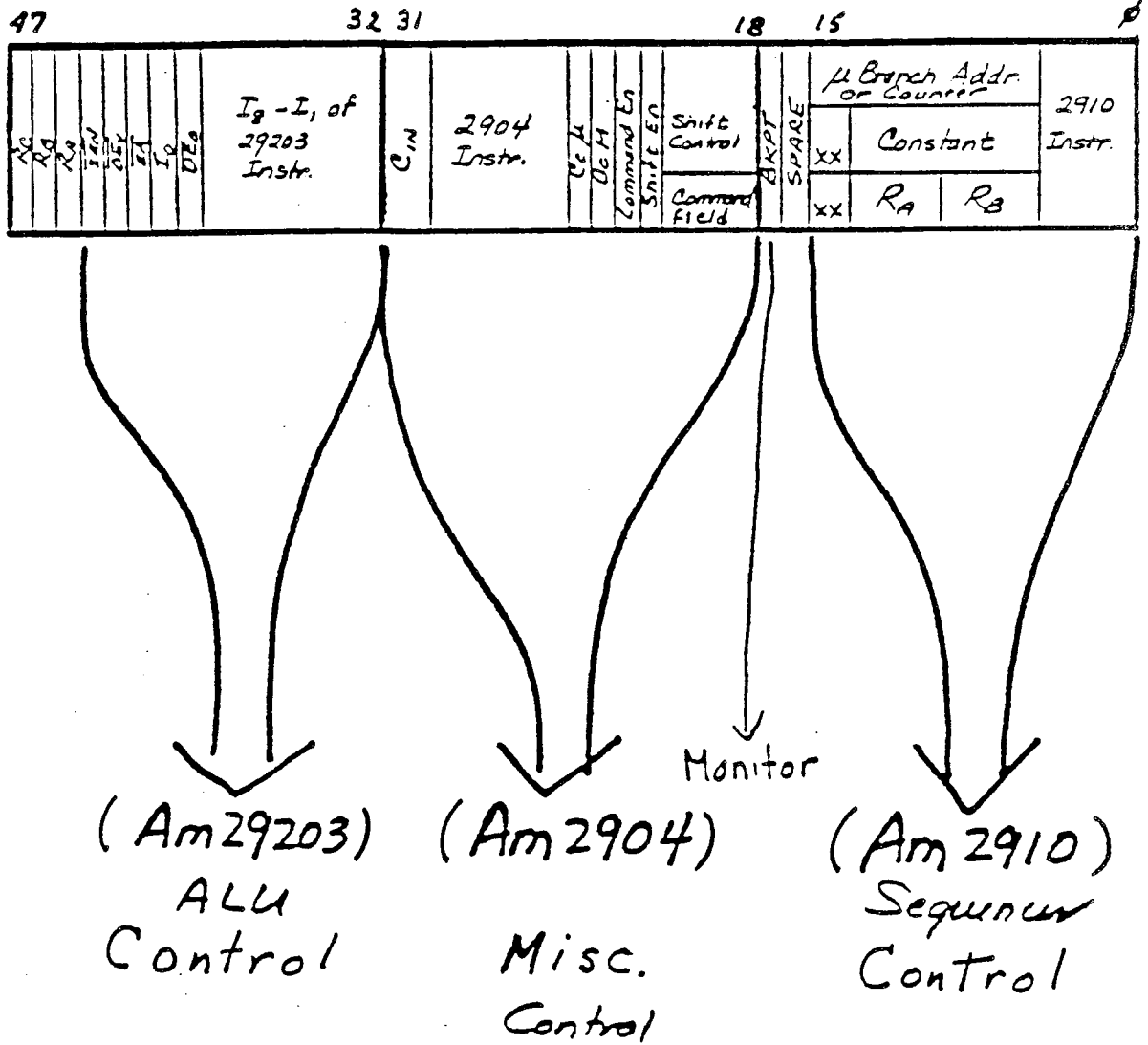
6. Implement and exercise the unsigned multiply operation with special functions. Use registers R0 and R1 for the augend and addend respectively, and registers R3 and R4 for storing the result.



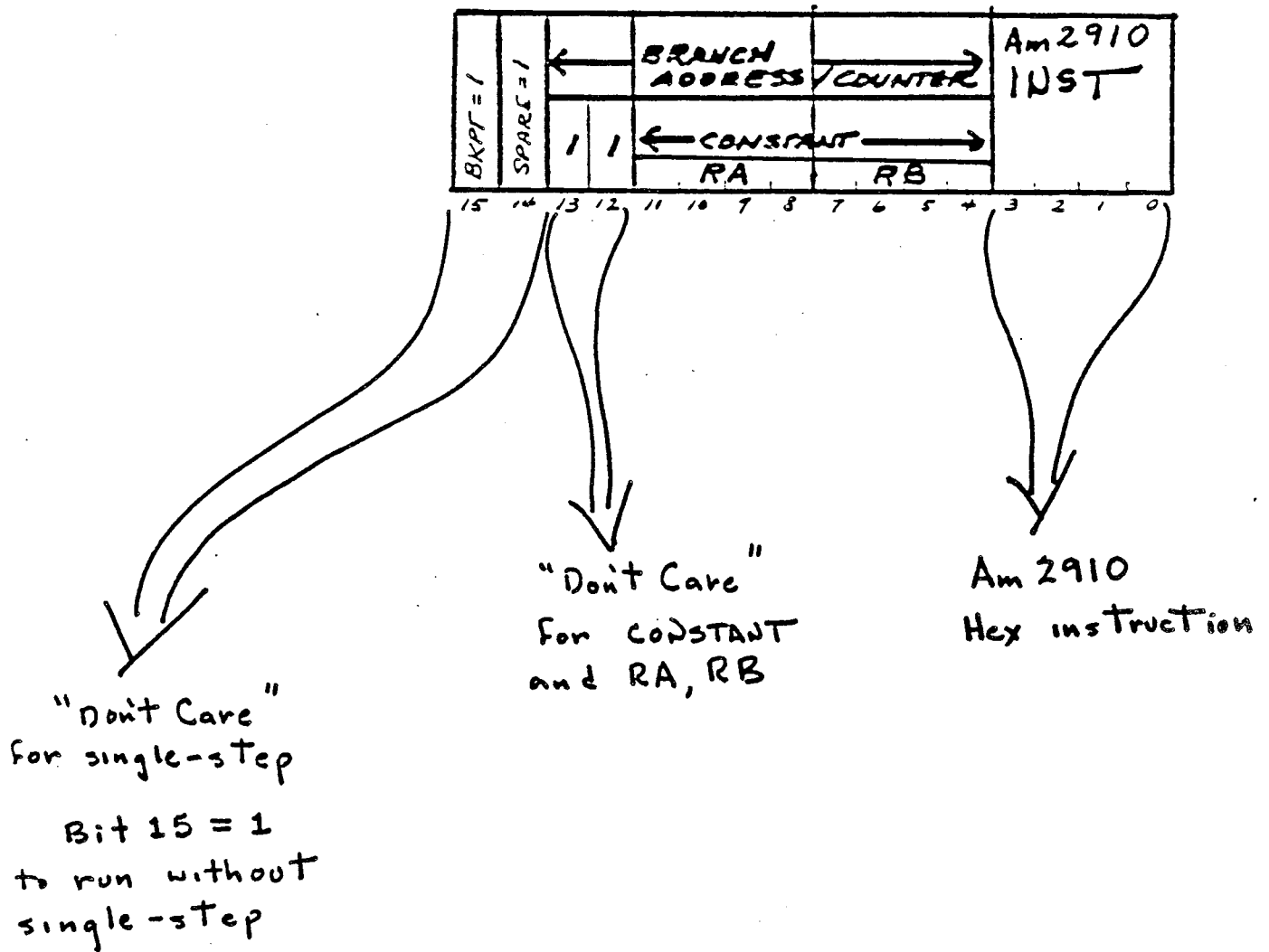
## APPENDIX A

### EVALUATION BOARD FIELD DEFINITIONS

# Evolution Board - Microword Format

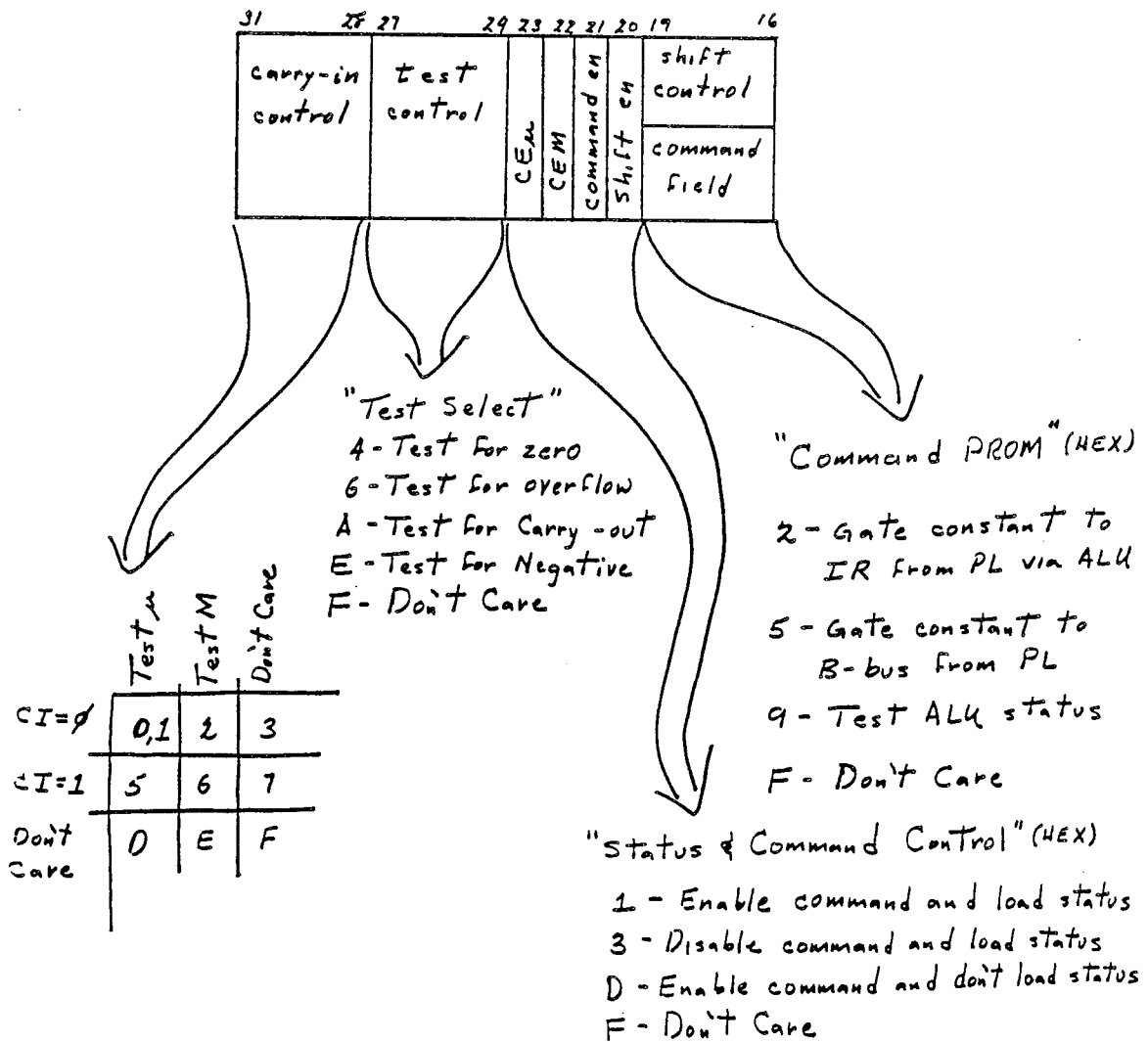


# Bits 15-0 Sequencer Control (Am2910)



# Bits 31-16 Misc. Control (Am 2904)

## "A Beginning"



## Carry-In Control Multiplexer Codes

I12	I11	I5	I3	I2	I1	C0
0	0	X	X	X	X	0
0	1	X	X	X	X	1
1	0	X	X	X	X	Cx
1	1	0	0	X	X	uC
1	1	0	X	1	X	uC
1	1	0	X	X	1	uC
1	1	0	1	0	0	uC*
1	1	1	0	X	X	MC
1	1	1	X	1	X	MC
1	1	1	X	X	1	MC
1	1	1	1	0	0	MC*

## STATUS (bits 29-24):

15-10 OCTAL	CEu*=0 MICRO	CEM*=0 MACRO	BOTH	Action if OEY*=0
00	MSRTOUSR	YTOMSR	YMSRUSR	UTOY
01	SETUSR	SETMSR	SETREGS	(00)
02	MSRTOUSR	USRTOMSR	SWAPREGS	(00)
03	RESETUSR	RESETMSR	RESETREGS	(00)
04	(20)	SWAPMCMO	?	(00)
05	(20)	INVERTMSR	?	(00)
06	IRETOVR1	(20)	?	(00)
(07)	(06)	(06)	?	(00)
10	RESETUZ	(30)	?	(00)
11	SETUZ	(30)	?	(00)
12	RESETUC	(20)	?	(00)
13	SETUC	(20)	?	(00)
14	RESETUN	(20)	?	(00)
15	SETUN	(20)	?	(00)
16	RESETUO	(20)	?	(00)
17	SETUO	(20)	?	(00)
20	ITOUSR	ITOMSR	ITOREGS	(00)
21-27	(20)	(20)	(20)	(00)
30	IWITHUC*	IWITHMC*	IWITHC*	(00)
31	(30)	(30)	(30)	(00)
32-37	(20)	(20)	(20)	(00)
40-47	(20)	(20)	(20)	MTOY
50-51	(30)	(30)	(30)	(40)
52-57	(20)	(20)	(20)	(40)
60-67	(20)	(20)	(20)	ITOY
70-71	(30)	(30)	(30)	(60)
72-77	(20)	(20)	(20)	(60)

TEST (bits 29-24):
--------------------

I5-I4	I3-I0		
OCTAL	HEX	MNEMONIC	TEST
1	4	UZ	Micro Zero
2	4	MZ	Macro Zero
3	4	IZ	I-bus Zero
1	5	UZ*	Micro Not-Zero
2	5	MZ*	Macro Not-Zero
3	5	IZ*	I-bus Not-Zero
1	6	UOVR	Micro Overflow
2	6	MOVR	Macro Overflow
3	6	IOVR	I-bus Overflow
1	7	UOVR*	Micro Not-Overflow
2	7	MOVR*	Macro Not-Overflow
3	7	IOVR*	I-bus Not-Overflow
1	A	UC	Micro Carry
2	A	MC	Macro Carry
3	A	IC	I-bus Carry
1	B	UC*	Micro Not-Carry
2	B	MC*	Macro Not-Carry
3	B	IC*	I-bus Not-Carry
1	E	UN	Micro Negative
2	E	MN	Macro Negative
3	E	IN	I-bus Negative
1	F	UN*	Micro Not-Negative
2	F	MN*	Macro Not-Negative
3	F	IN*	I-bus Not-Negative

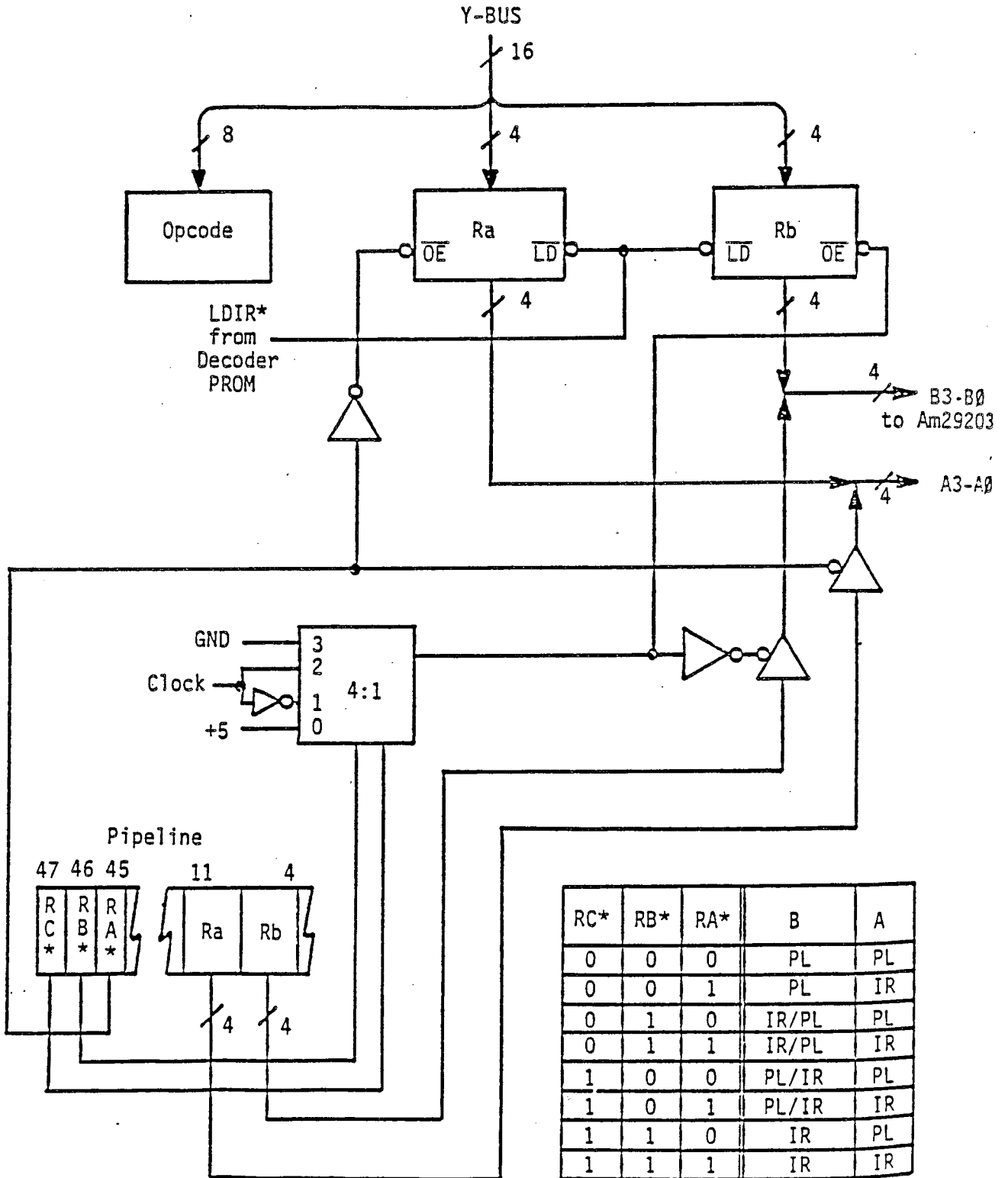
## 8-13. Test Control Fields &amp; Mnemonics

Am2904 TABLE 7. SHIFT LINKAGE MULTIPLEXER INSTRUCTION CODES.

$I_{10}$	$I_9$	$I_8$	$I_7$	$I_6$	$M_C$	RAM	Q	$SIO_0$	$SIO_n$	$QIO_0$	$QIO_n$	Loaded into $M_C$
0	0	0	0	0	<input type="checkbox"/>	MSB LSB MSB LSB		Z	0	Z	0	
0	0	0	0	1	<input type="checkbox"/>	MSB LSB MSB LSB		Z	1	Z	1	
0	0	0	1	0	<input type="checkbox"/>	MSB LSB MSB LSB		Z	0	Z	$M_N$	$SIO_0$
0	0	0	1	1	<input type="checkbox"/>	MSB LSB MSB LSB		Z	1	Z	$SIO_0$	
0	0	1	0	0	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$M_C$	Z	$SIO_0$	
0	0	1	0	1	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$M_N$	Z	$SIO_0$	
0	0	1	1	0	<input type="checkbox"/>	MSB LSB MSB LSB		Z	0	Z	$SIO_0$	
0	0	1	1	1	<input type="checkbox"/>	MSB LSB MSB LSB		Z	0	Z	$SIO_0$	$QIO_0$
0	1	0	0	0	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$SIO_0$	Z	$QIO_0$	$SIO_0$
0	1	0	0	1	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$M_C$	Z	$QIO_0$	$SIO_0$
0	1	0	1	0	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$SIO_0$	Z	$QIO_0$	
0	1	0	1	1	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$I_C$	Z	$SIO_0$	
0	1	1	0	0	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$M_C$	Z	$SIO_0$	$QIO_0$
0	1	1	0	1	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$QIO_0$	Z	$SIO_0$	$QIO_0$
0	1	1	1	0	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$I_N \oplus I_{OVR}$	Z	$SIO_0$	
0	1	1	1	1	<input type="checkbox"/>	MSB LSB MSB LSB		Z	$QIO_0$	Z	$SIO_0$	
1	0	0	0	0	<input type="checkbox"/>	MSB LSB MSB LSB		0	Z	0	Z	$SIO_n$
1	0	0	0	1	<input type="checkbox"/>	MSB LSB MSB LSB		1	Z	1	Z	$SIO_n$
1	0	0	1	0	<input type="checkbox"/>	MSB LSB MSB LSB		0	Z	0	Z	
1	0	0	1	1	<input type="checkbox"/>	MSB LSB MSB LSB		1	Z	1	Z	
1	0	1	0	0	<input type="checkbox"/>	MSB LSB MSB LSB		$QIO_n$	Z	0	Z	$SIO_n$
1	0	1	0	1	<input type="checkbox"/>	MSB LSB MSB LSB		$QIO_n$	Z	1	Z	$SIO_n$
1	0	1	1	0	<input type="checkbox"/>	MSB LSB MSB LSB		$QIO_n$	Z	0	Z	
1	0	1	1	1	<input type="checkbox"/>	MSB LSB MSB LSB		$QIO_n$	Z	1	Z	
1	1	0	0	0	<input type="checkbox"/>	MSB LSB MSB LSB		$SIO_n$	Z	$QIO_n$	Z	$SIO_n$
1	1	0	0	1	<input type="checkbox"/>	MSB LSB MSB LSB		$M_C$	Z	$QIO_n$	Z	$SIO_n$
1	1	0	1	0	<input type="checkbox"/>	MSB LSB MSB LSB		$SIO_n$	Z	$QIO_n$	Z	
1	1	0	1	1	<input type="checkbox"/>	MSB LSB MSB LSB		$M_C$	Z	0	Z	
1	1	1	0	0	<input type="checkbox"/>	MSB LSB MSB LSB		$QIO_n$	Z	$M_C$	Z	$SIO_n$
1	1	1	0	1	<input type="checkbox"/>	MSB LSB MSB LSB		$QIO_n$	Z	$SIO_n$	Z	$SIO_n$
1	1	1	1	0	<input type="checkbox"/>	MSB LSB MSB LSB		$QIO_n$	Z	$M_C$	Z	
1	1	1	1	1	<input type="checkbox"/>	MSB LSB MSB LSB		$QIO_n$	Z	$SIO_n$	Z	

Notes: 1. Z = High impedance (outputs off) state. 2. Outputs enabled and  $M_C$  loaded only if  $\overline{SE}$  is LOW. 3. Loading of  $M_C$  from  $I_{10-6}$  overrides control from  $I_{5-0}$ ,  $\overline{CE}_U$ ,  $\overline{EC}$ .





Am29203 Register Address Selection

## Am29203 Source Codes &amp; Mnemonics

Mnemonic	↑2	40
	Ea*, I0, 0Eb*	
RAMAB	0	
RAMADB	1	
RAMAQ	2++	
RAMAQ	3++	
DARAMB	4	
DADB	5	
DAQ	6++	
DAQ	7++	

++ I0 HIGH

35 - 32  

FUNCT
-------

DEF File for Am29203 ALU Basic Functions

```

SUBR:      EQU      H#1      ;F = S - R - 1 + Cin
SUBS:      EQU      H#2      ;F = R - S - 1 + Cin
ADD:       EQU      H#3      ;F = R + S + Cin
INCRS:     EQU      H#4      ;F = S + Cin
INCRSNON:  EQU      H#5      ;F = .S + Cin

```

```

NOTRS:     EQU      H#9      ;Fi = .Ri AND Si
EXNOR:     EQU      H#A      ;Fi = Ri EXNOR Si
EXOR:      EQU      H#B      ;Fi = Ri EXOR Si
AND:       EQU      H#C      ;Fi = Ri AND Si
NOR:       EQU      H#D      ;Fi = Ri NOR Si
NAND:      EQU      H#E      ;Fi = Ri NAND Si
OR:        EQU      H#F      ;Fi = Ri OR Si

```

; \*\*\* The following require that RAMAQ or DAQ be the source:

```

HIGH:      EQU      H#0      ;Fi = HIGH
INCRR:     EQU      H#6      ;F = R + Cin
INCRNON    EQU      H#7      ;F = .R + Cin
LOW:       EQU      H#8      ;Fi = LOW

```