MODULE 5: Assembly Language + Processor Control + Examples

Lecture 5.3
Processor Control

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering

Virginia Tech

Lecture 5.3 Objectives

- Compare and contrast microprogrammed and hardwired approaches to processor control
- Identify the control signals used in the MARIE datapath
- Given a MARIE instruction, produce the sequence of control signals that result in the execution of the instruction
- Interpret segments of a microprogram that implement the MARIE ISA

Microarchitecture (1)

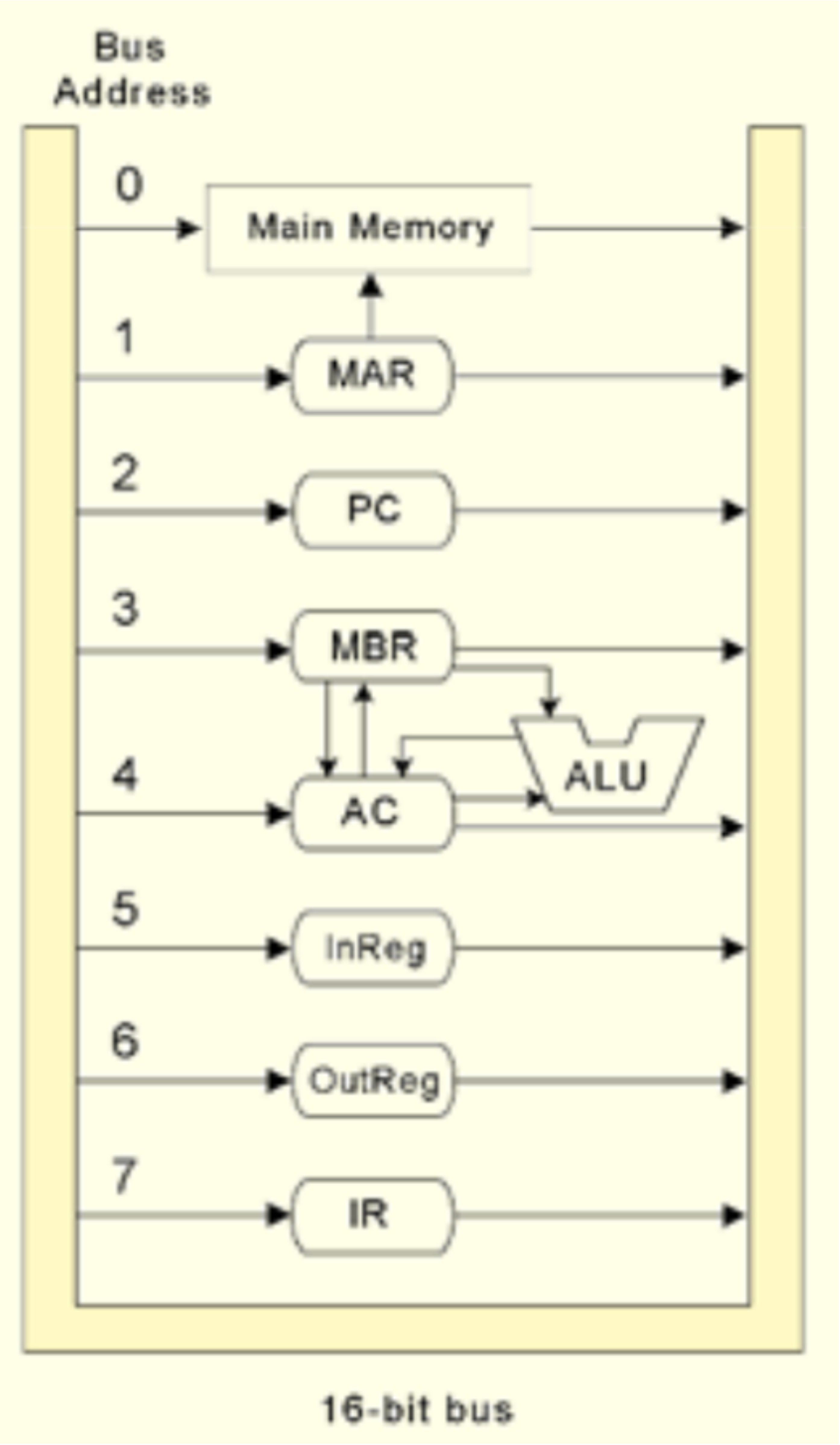
- The instruction set architecture (ISA) is the view of the machine as seen by the assembly language programmer
  - Instructions, registers, memory organization, input/output organization
- The microarchitecture implements the ISA
  - Control unit of the central processing unit (CPU)
  - Functional units, such as the arithmetic logic unit (ALU)
  - Registers visible to the assembly language programmer
  - Additional registers needed for the control unit to implement the ISA

Microarchitecture (2)

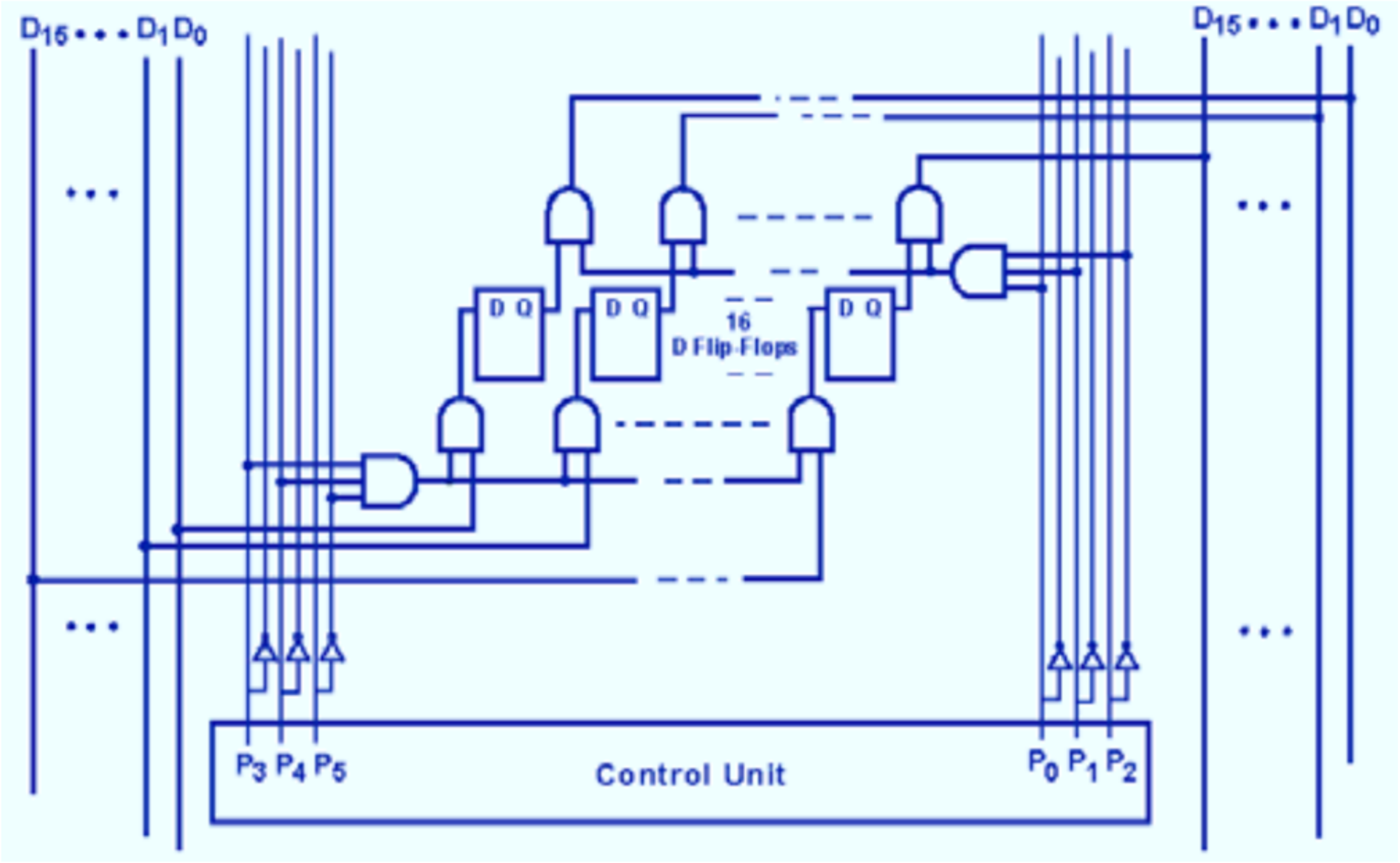
- A given microarchitecture is just one implementation for an ISA — there may be more than one microarchitecture for an ISA
  - For example, competing implementations of the Pentium ISA
- Basic microarchitecture approaches
  - Microprogrammed control: A highly-specialized program that implements the ISA is stored in read-only memory
  - Hardwired control: The microarchitecture is a direct hardware implementation (digital logic)

MARIE Datapath

- Each of MARIE's registers and main memory have a unique address along the datapath
- The addresses take the form of signals issued by the control unit
- One set of signals P_0, P_1, P_2, controls reading from memory or a register, and the other set P_3, P_4, P_5, controls writing to memory or a register



Example: MBR Implementation



- Register is enabled for reading when P_0 and P_1 are high and for writing when P_3 and P_4 are high

MARIE Control Signals

- Register controls: P_0 through P_5
  - Reading is enabled by P_2 P_1 P_0 (P_2 is the MSB)
  - Writing is enabled by P_5 P_4 P_3 (P_5 is the MSB)
- ALU controls: A_0 through A_1
  - Operations: add, subtract, clear, do nothing
- Timing: T_0 through T_7 and counter reset C_r
  - Cycle counter coordinates the activities (that are part of the execution of a single instruction) taking place at each clock cycle
  - Cycle counter reset signal resets the counter to get ready for the next instruction

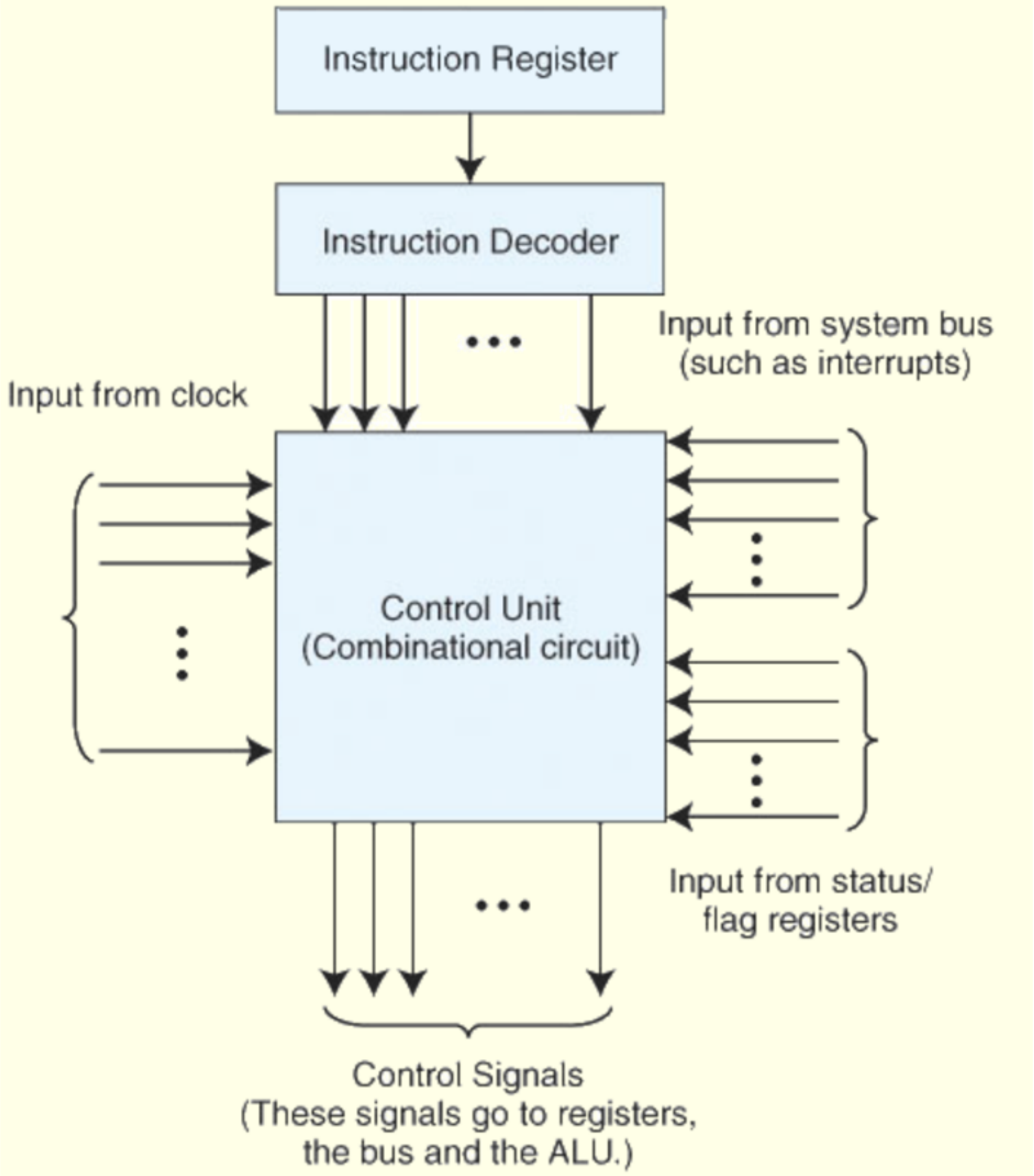
CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- State the control signals in the MARIE ISA for the Register Controls, ALU Controls, and Timing Controls

If you have any difficulties, please review the lecture video before continuing.

Hardwired Control Unit



Add Instruction Signal Sequence (1)

Add X

1

$$\begin{array}{rl} P_0\, P_1\, P_2\, P_3\, T_0 : & MAR \leftarrow X \\ P_3\, P_4\, T_1 : & MBR \leftarrow M[MAR] \\ A_0\, P_0\, P_1\, P_5\, T_2 : & AC \leftarrow AC + MBR \\ C_r\, T_3 : & \text{[Reset clock cycle counter]} \end{array}$$

1. Must bring X from the IR (datapath address of 7), so to enable reading the IR must raise P_0 P_1 P_2
   To enable writing to the MAR (datapath address of 1) must raise P_3
   Raise T_0 to indicate the first clock cycle for this instruction

Add Instruction Signal Sequence (2)

Add X

$\boxed{2}$

$P_0\ P_1\ P_2\ P_3\ T_0$:	$MAR \leftarrow X$
$P_3\ P_4\ T_1$:	$MBR \leftarrow M[MAR]$
$A_0\ P_0\ P_1\ P_5\ T_2$:	$AC \leftarrow AC + MBR$
$C_r\ T_3$:	[Reset clock cycle counter]

2. Must bring data from memory (datapath address of 0), so to enable reading the memory P_0 P_1 P_2 must remain low
   To enable writing to the MBR (datapath address of 3) must raise P_4 P_3
   Raise T_1 to indicate the second clock cycle for this instruction

Add Instruction Signal Sequence (3)

Add X

$\boxed{3}$

$P_0\,P_1\,P_2\,P_3\,T_0$:	MAR $\leftarrow$ X
$P_3\,P_4\,T_1$:	MBR $\leftarrow$ M[MAR]
$A_0\,P_0\,P_1\,P_5\,T_2$:	AC $\leftarrow$ AC + MBR
$C_r\,T_3$:	[Reset clock cycle counter]

3. To specify the add ALU operation, raise A_0
   To read from the MBR (datapath address of 3) into the ALU, raise $P_0\ P_1$
   To write to the AC (datapath address of 4) raise P_5
   Raise T_2 to indicate the third clock cycle for this instruction

Add Instruction Signal Sequence (4)

Add X

$P_0\,P_1\,P_2\,P_3\,T_0$: $MAR \leftarrow X$

$P_3\,P_4\,T_1$: $MBR \leftarrow M[MAR]$

$A_0\,P_0\,P_1\,P_5\,T_2$: $AC \leftarrow AC + MBR$

4 $C_r\,T_3$: [Reset clock cycle counter]

4. To reset the clock cycle counter, raise C_r
   Raise T_3 to indicate the fourth clock cycle for this instruction

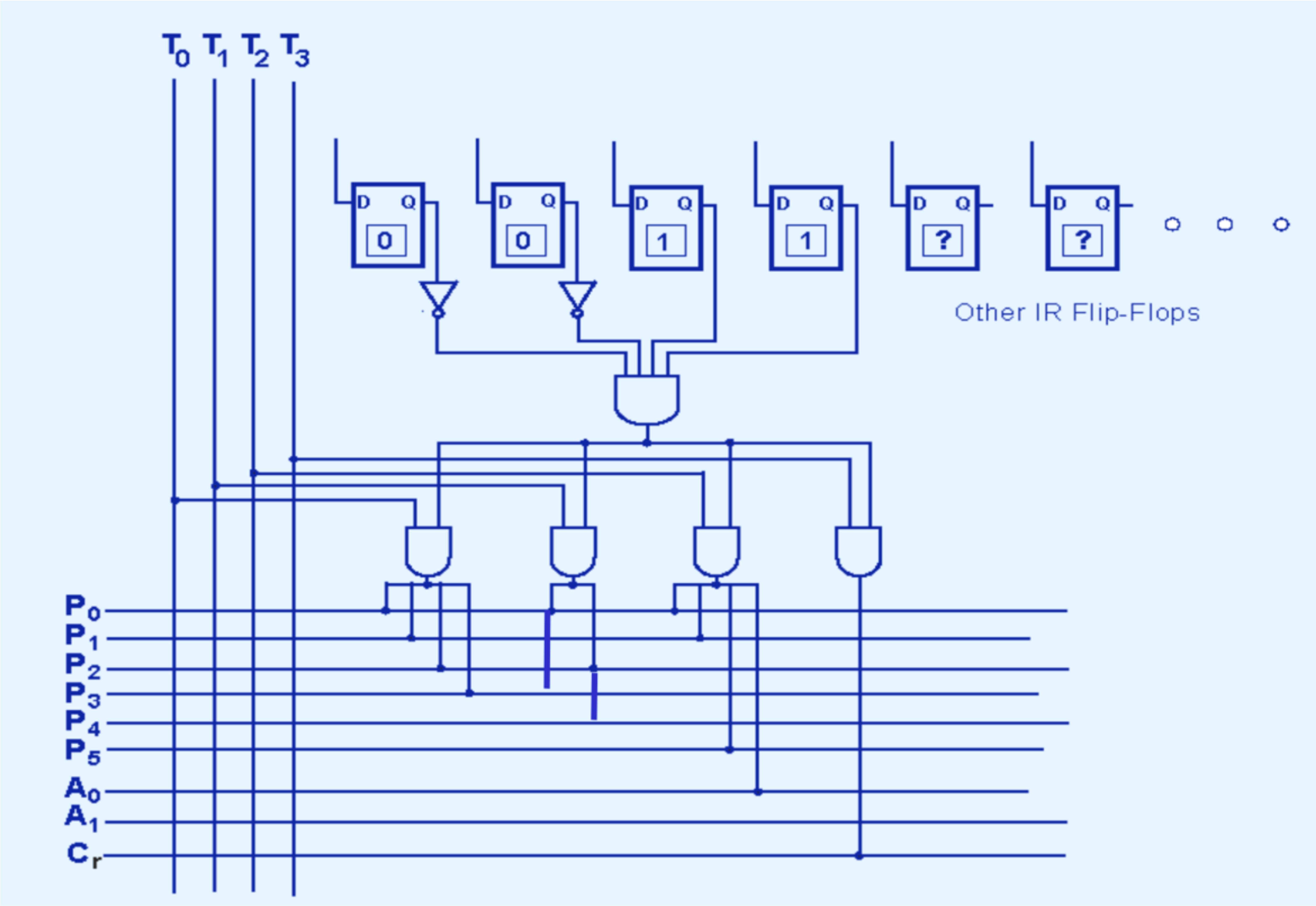
CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Step through by hand the hardwired control signals for the Add instruction

If you have any difficulties, please review the lecture video before continuing.

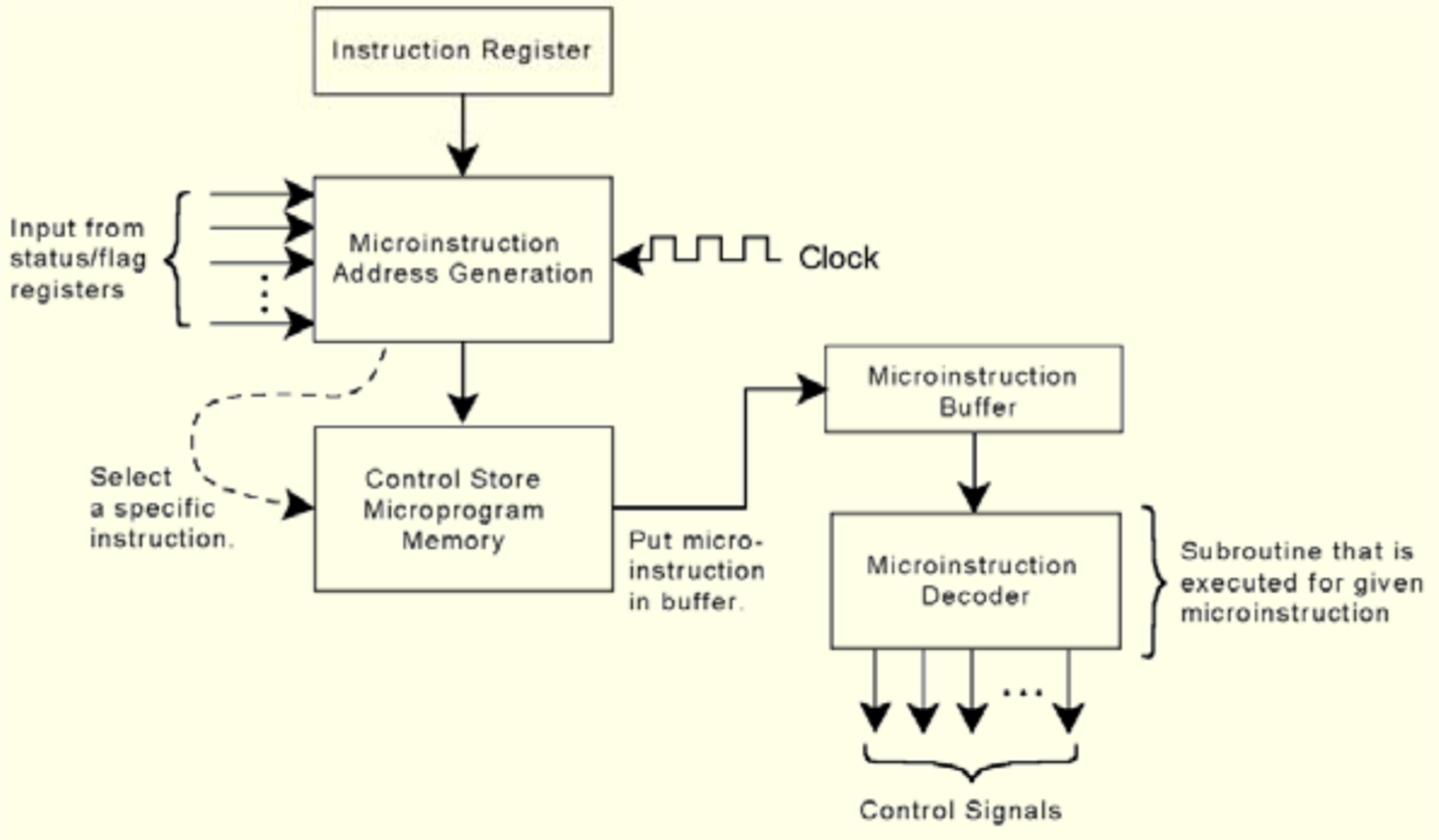
Add Instruction: Hardwired Logic



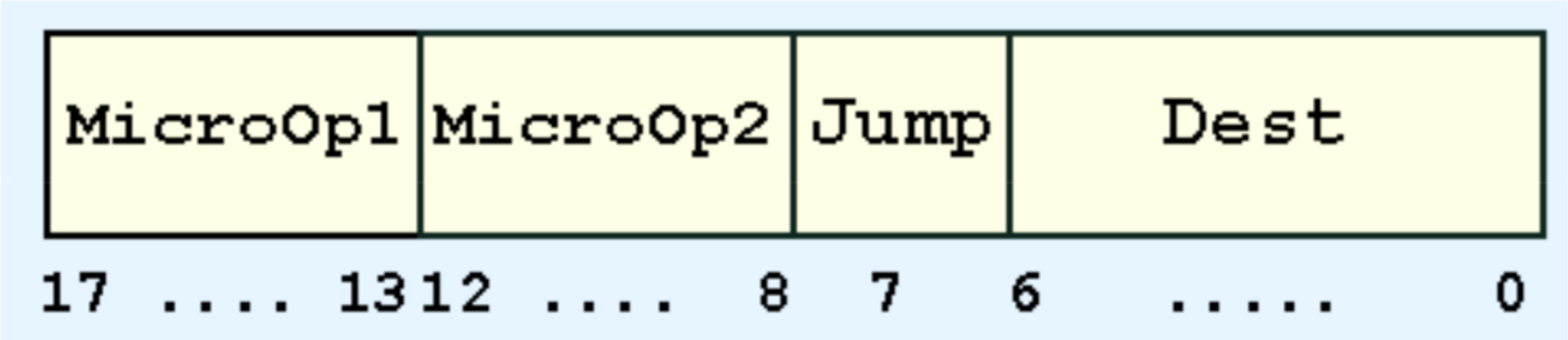
Microprogrammed Control

- In microprogrammed control, instruction microcode produces control signal changes
- Machine instructions are the input for a microprogram that converts the 1s and 0s of an instruction into control signals
- The microprogram is stored in firmware, which is also called the control store
- A microcode instruction is retrieved during each clock cycle
- The sequence of signals described for the Add instruction is the same whether we're using hardwired logic or microprogrammed control

Microprogrammed Control Unit



Microinstruction Format



- MicroOp1 and MicroOp2 contain binary codes for each instruction
- Jump is a single bit indicating that the value in the Dest field is a valid address and should be placed in the microsequencer

MARIE Microoperation Codes

MicroOp Code	Microoperation	MicroOp Code	Microoperation
00000	NOP	01100	MBR ← M[MAR]
00001	AC ← 0	01101	OutREG ← AC
00010	AC ← AC – MBR	01110	PC ← IR[11-0]
00011	AC ← AC + MBR	01111	PC ← MBR
00100	AC ← InREG	10000	PC ← PC + 1
00101	IR ← M[MAR]	10001	If AC = 00
00110	M[MAR] ← MBR	10010	If AC > 0
00111	MAR ← IR[11-0]	10011	If AC < 0
01000	MAR ← MBR	10100	If IR[11-10] = 00
01001	MAR ← PC	10101	If IR[11-10] = 01
01010	MAR ← X	10110	If IR[11-10] = 10
01011	MBR ← AC	10111	If IR[15-12] = MicroOp2[4-1]

MARIE Microprogram: Fetch/Execute

Address	MicroOp 1	MicroOp 2	Jump	Dest
0000000	MAR ← PC	NOP	0	0000000
0000001	IR ← M[MAR]	NOP	0	0000000
0000010	PC ← PC + 1	NOP	0	0000000
0000011	MAR ← IR[11-0]	NOP	0	0000000
0000100	If IR[15-12] = MicroOp2[4-1]	00000	1	0100000
0000101	If IR[15-12] = MicroOp2[4-1]	00010	1	0100111
0000110	If IR[15-12] = MicroOp2[4-1]	00100	1	0101010
0000111	If IR[15-12] = MicroOp2[4-1]	00110	1	0101100
0001000	If IR[15-12] = MicroOp2[4-1]	01000	1	0101111
...	...	...	...	...

Microprogrammed Control Tradeoffs

- A microprogrammed control unit works like a system-in-miniature
- Microinstructions are fetched, decoded, and executed in the same manner as regular instructions
- This extra level of instruction interpretation is what makes microprogrammed control slower than hardwired control
- The advantages of microprogrammed control are that it can support very complicated instructions and only the microprogram needs to be changed if the instruction set changes (or an error is found)

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Step through the microprogram instructions for the fetch/execute cycle
- Compare and contrast hardwired control versus microprogrammed control

If you have any difficulties, please review the lecture video before continuing.

Summary

- In microprogrammed control, a highly-specialized program that implements the ISA is stored in read-only memory
  - Machine instructions are the input for a microprogram stored in firmware that converts the 1s and 0s of an instruction into control signals
- In hardwired control, the microarchitecture is a direct hardware implementation (digital logic)
- MARIE control signals include signals to read from and write to registers and memory, to specify an ALU operation, and to control timing

MODULE 5: Assembly Language + Processor Control + Examples

Lecture 5.3
Processor Control

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech