SECR2033
Computer Organization and Architecture

Lecture slides prepared by “Computer Organization and Architecture”, 9/e, by William Stallings, 2013.

Module 5
Central Processing Unit (CPU)

Objectives:

- To learn the components common to every CPU.
- Be able to explain how each component in CPU contributes to instruction cycle.
- To understand the concept of pipelining in CPU execution.
- Be able to understand micro-operations basis for the design and implementation of the control unit.

Module 5
Central Processing Unit (CPU)

5.1 Processor Organization
5.2 Register Organization
5.3 Instruction Cycles

5.4 Instruction Pipelining

5.5 Control Unit Operation
5.6 Microprogrammed Control

5.7 Summary

Module 5c
Central Processing Unit (CPU)

- Overview
- Micro-Operation (1)
- Control of the Processor

Let's recall from Module 1

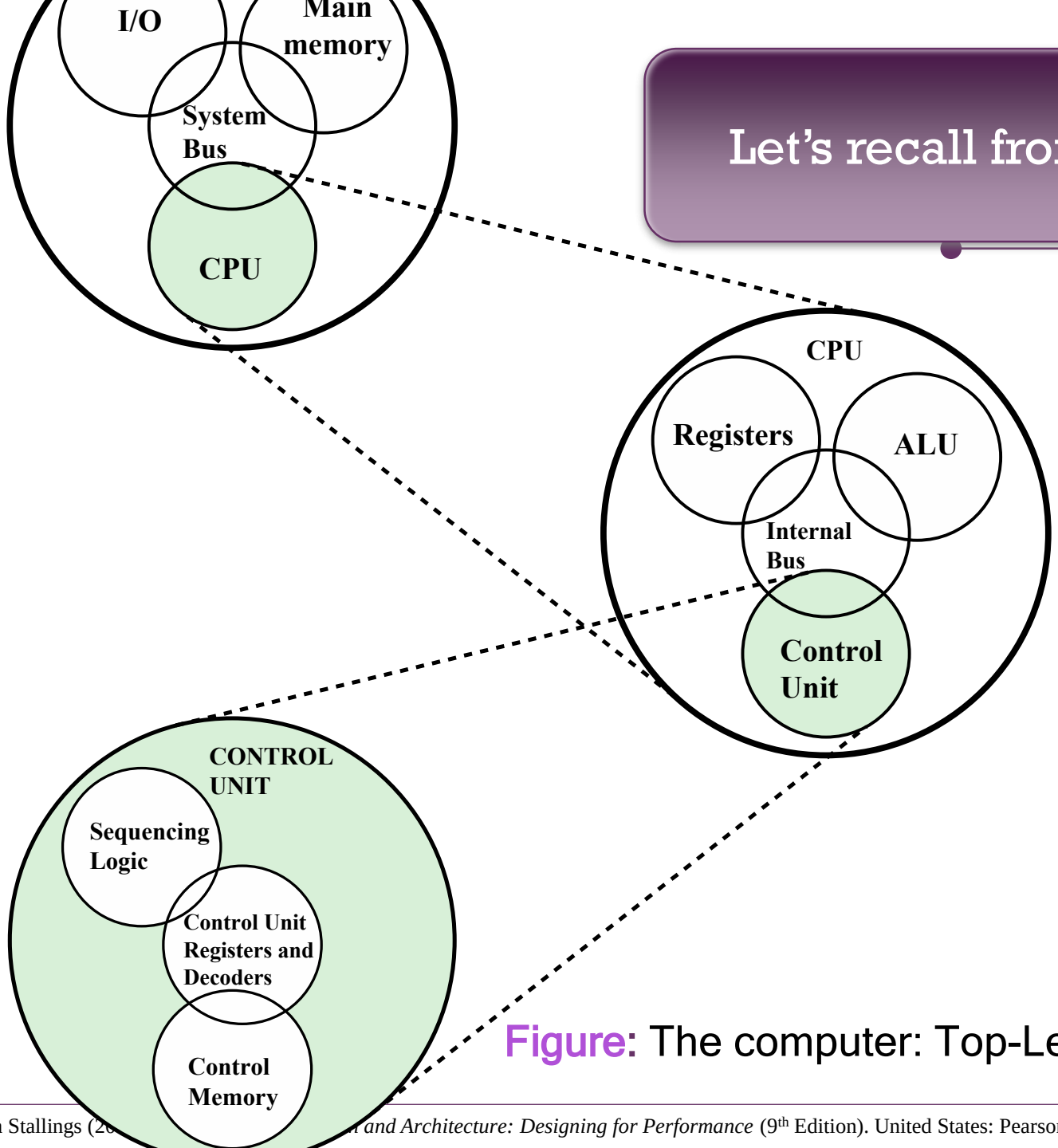


Figure: The computer: Top-Level structure

Minimal internal memory, consisting of a set of storage locations, called registers.

Let's recall from slide 7 Module 5a

ALU does the actual computation or processing of data

Control unit controls the movement of data and instructions into / out of the processor and controls the operation of the ALU

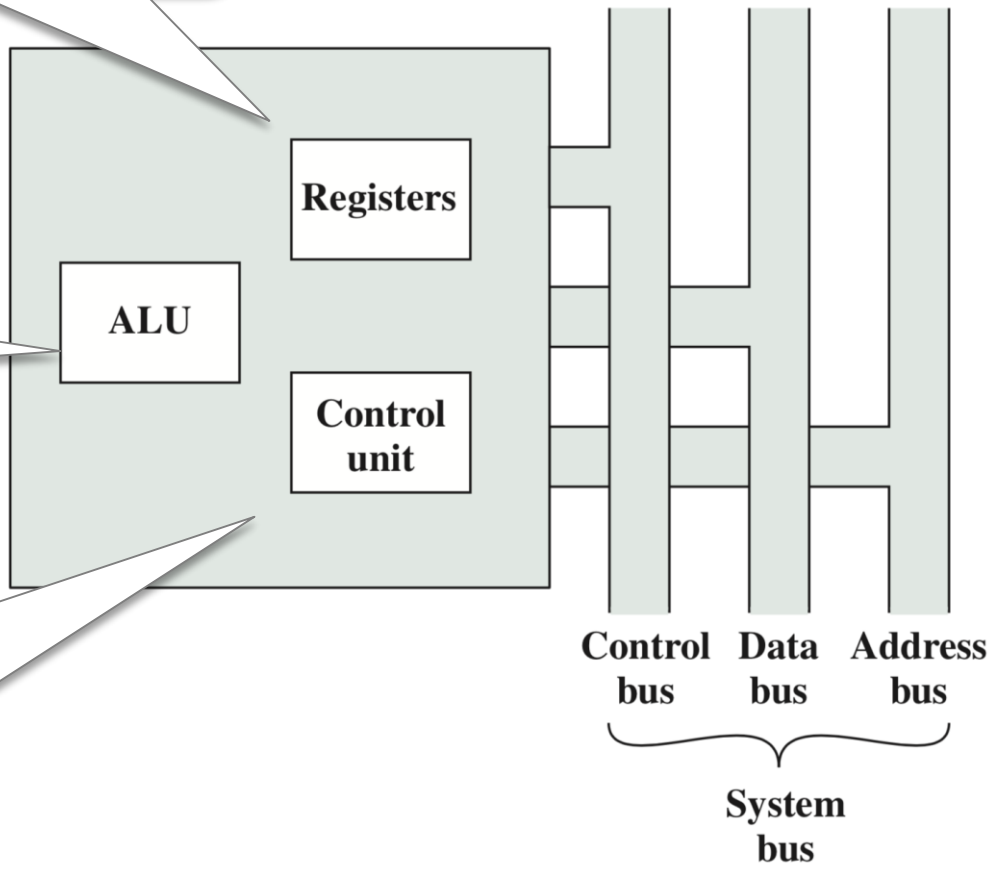


Figure: The CPU with the System Bus.

5 Overview

- Here, we have to know the external interfaces, usually through a bus, and how interrupts are handled.

- Six items might be termed the functional requirements for a processor should do and being controlled :

1. Operations (opcodes)	4. I/O module interface
2. Addressing modes	5. Memory module interface
3. Registers	6. Interrupts

5 Control of the Processor

- The basis for the design and implementation of the control unit are based on the definition of a functional requirements.
- Three-step process leads to a characterization of the control unit (CU):

1) Define the **basic elements** of the processor.
2) Describe the **micro-operations** (μ**OP**) that the processor performs.
3) Determine the **functions** that the Control Unit (CU) must perform to cause the micro-operations to be performed.

(1) Basic Elements of the Processor

- The basic functional elements of the processor are the following :

  - ALU – the functional essence of the computer.
  - CU – causes operations to happen within the processor.
  - Registers – used to store data (also status) internal to the processor.
  - Internal data paths – used to move data between registers and between register and ALU.
  - External data paths – link registers to memory and I/O modules, often by means of a system bus.

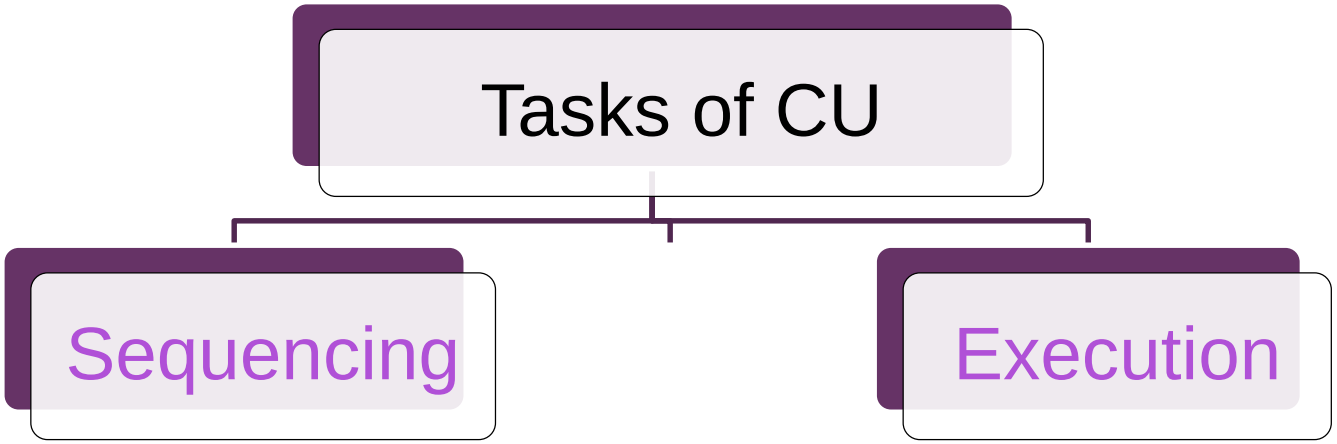
(2) Micro-Operation of that Processor Performs

- All micro-operations (μ**OP**) fall into one of the following categories :

  - Transfer data from one register to another register.
  - Transfer data from a register to an external interface (*e.g.* system bus).
  - Transfer data from an external interface to a register.
  - Perform an arithmetic or logic operation, using registers for input and output.

(3) Control Unit Functions

This is a functional description of what the Control Unit (CU) does → achieved through the use of control signals

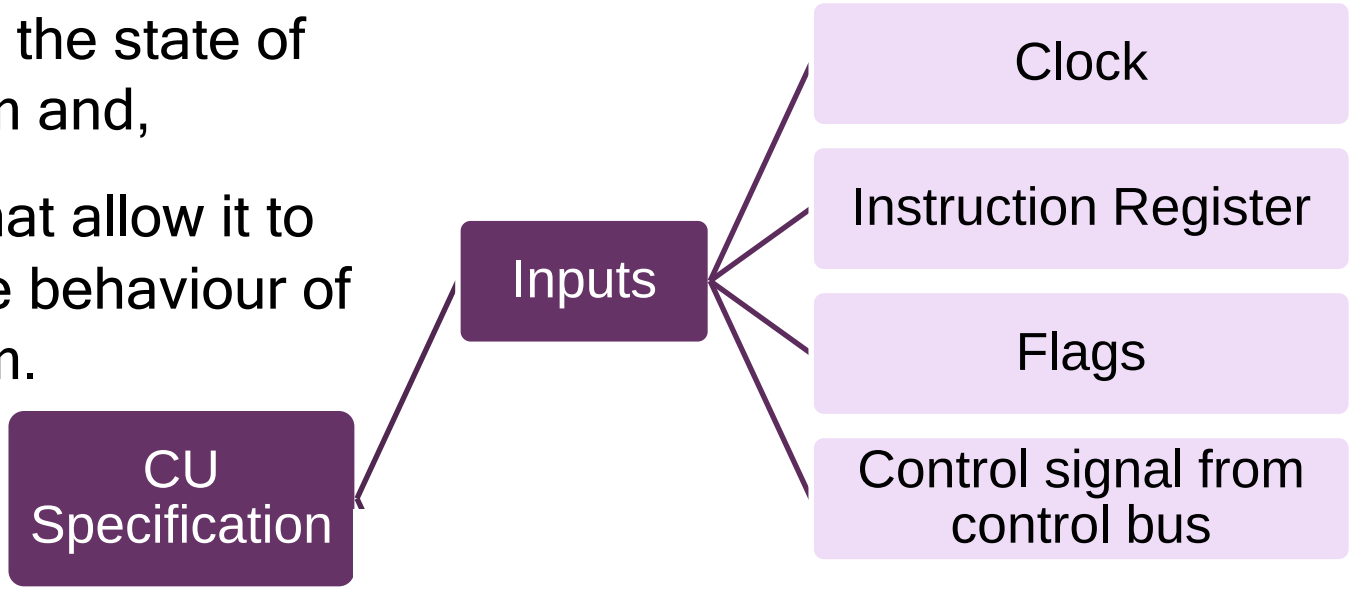


The CU causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed.

The CU causes each micro-operation to be performed.

Control Signals

- For the Control Unit (CU) to perform its function, it must have:
  - *Inputs* that allow it to determine the state of the system and,
  - *Outputs* that allow it to control the behaviour of the system.



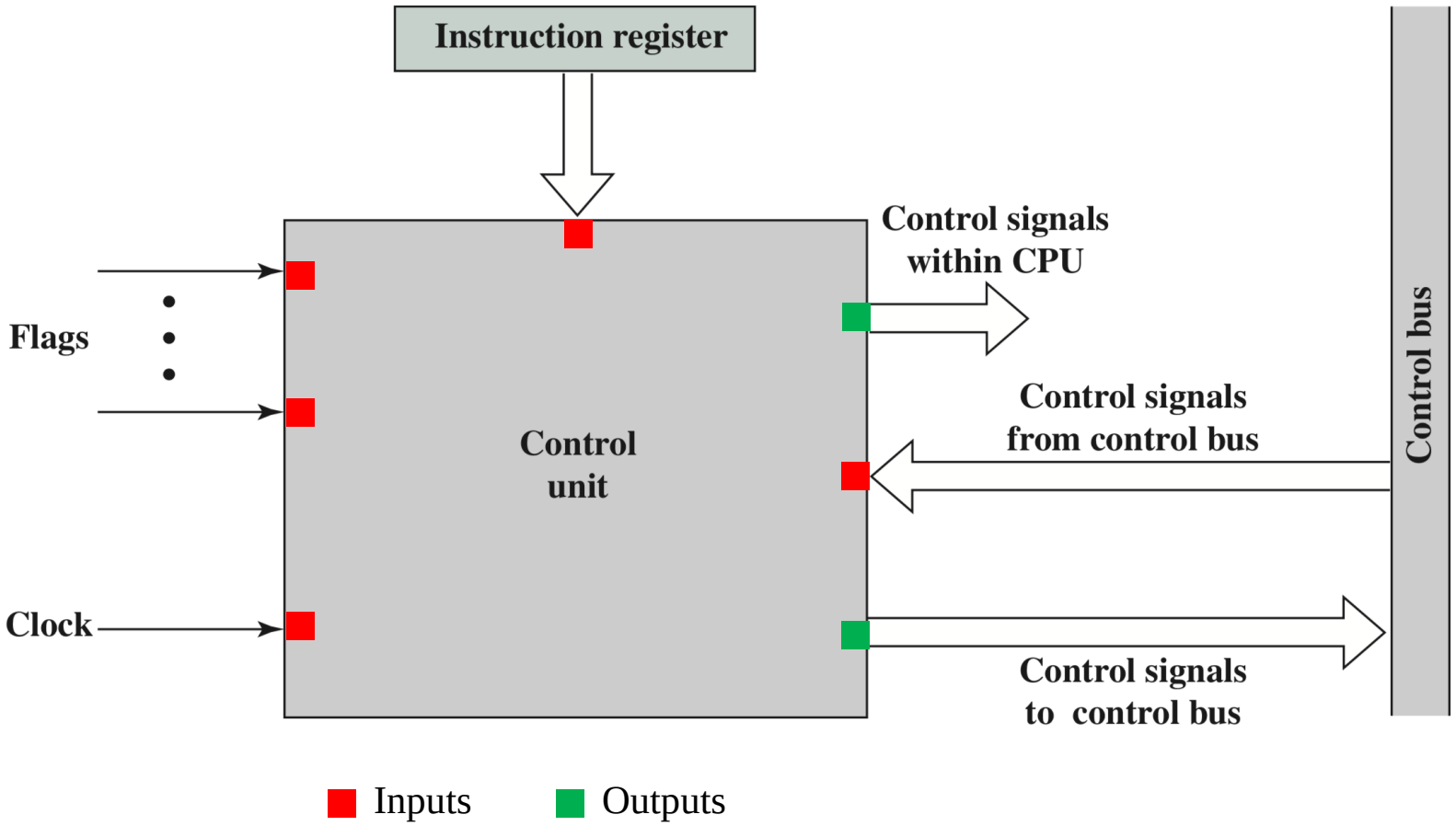


Figure: Block Diagram of the Control Unit

5

Table: The inputs of CU specification.

Inputs	Description
Clock	This is how the Control Unit (CU) "keeps time."
Instruction Register	The opcode and addressing mode of the current instruction are used to determine which micro-operations to perform during the execute cycle.
Flags	These are needed by the CU to determine the status of the processor and the outcome of previous ALU operations.
Control signal from control bus	The control bus portion of the system bus provides signals to the CU.

5

Table: The outputs of CU specification.

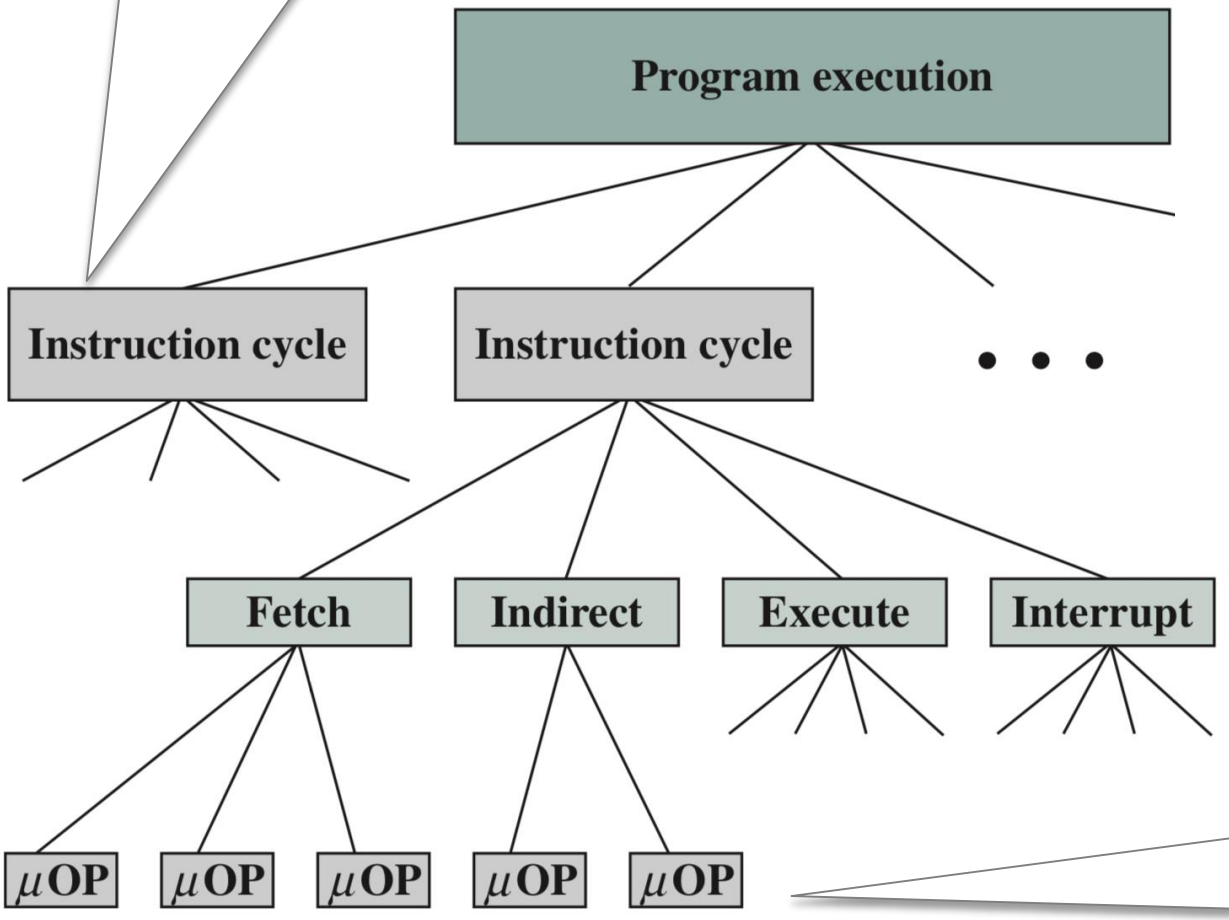
Outputs	Description
Control signal within the processor	These are two types: ❑ those that cause data to be moved from one register to another, and ❑ those that activate specific ALU functions
Control signals to control bus	These are also of two types: ❑ control signals to memory, and ❑ control signals to the I/O modules.

5 Micro-Operation (μOP)

- The operation of a computer, in executing a program, consists of a sequence of instruction cycles, with one machine instruction per cycle.
- Each instruction cycle is made up of a number of smaller units.
  - One subdivision that we found convenient is fetch, indirect, execute, and interrupt, with only *fetch* and *execute* cycles always occurring.
- Each of the smaller cycles involves a series of steps, each of which involves the processor registers.

Micro-Operation

*A program execution consists of the sequential execution of **instructions**.*



*Sub-**cycles***

Each sub-cycle involves one or more shorter operations → ***Micro-operations***

Figure: Constituent Elements of a Program Execution

Rules for Micro-Operations Grouping

- Proper sequence must be followed
  - MAR ← (PC) must precede MBR ← (memory)
- Conflicts must be avoided
  - Must not read and write same register at same time
  - MBR ← (memory) and IR ← (MBR) must not be in same cycle / during the same time unit

Execute Cycle

- Because of the variety of opcodes, there are a number of different sequences of micro-operations that can occur
- Instruction decoding
  - The control unit examines the opcode and generates a sequence of micro-operations based on the value of the opcode
- A simplified add instruction:
  - ADD R1, X (which adds the contents of the location X to register R1)

t1 : MAR ← (IR(Address))

t2 : MBR ← Memory

t3 : R1 ← (R1) + (MBR)

- In the first step the address portion of the IR is loaded into the MAR
- Then the referenced memory location is read
- Finally the contents of R1 and MBR are added by the ALU

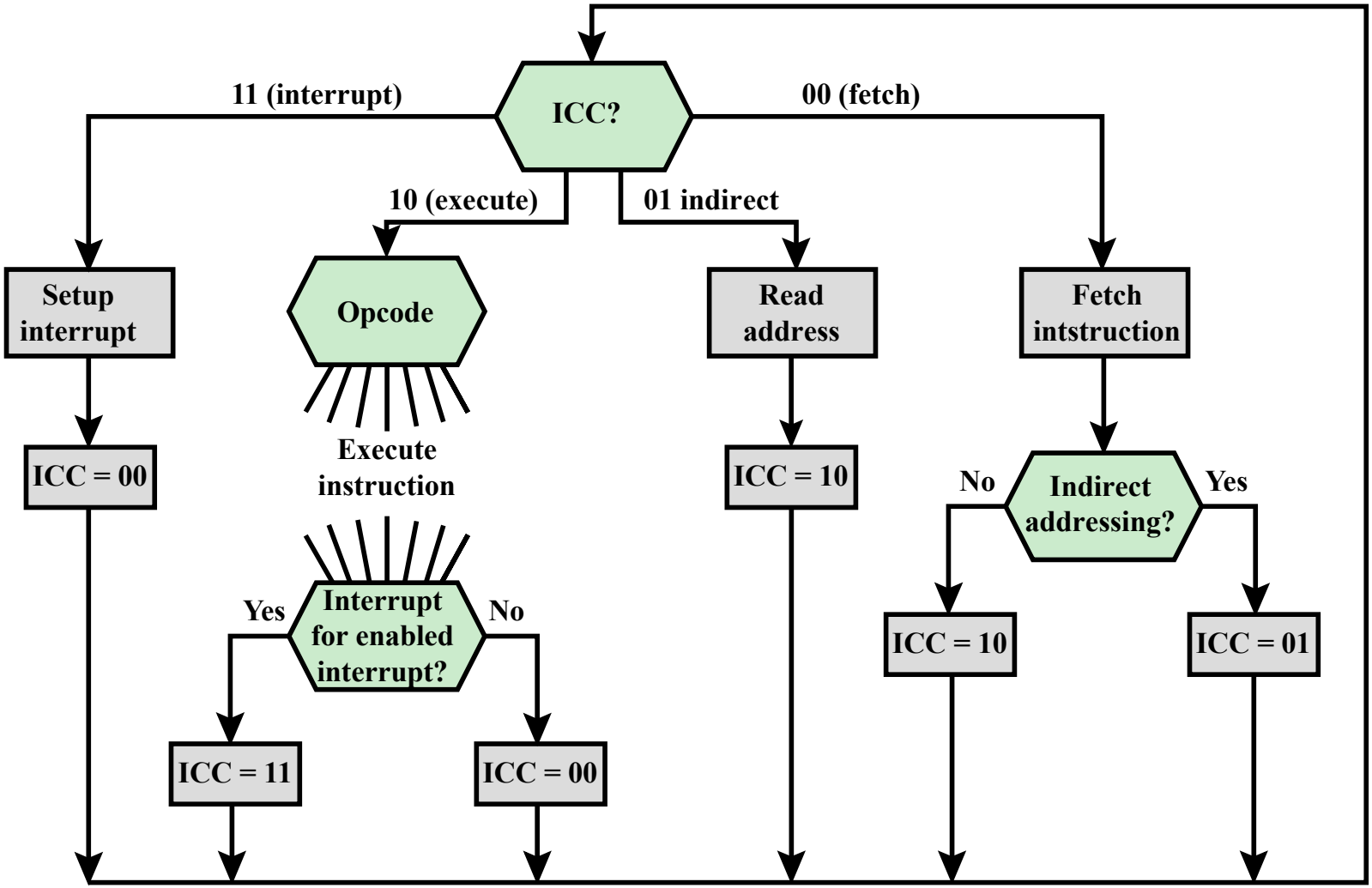


Figure 20.3 Flowchart for Instruction Cycle

MBR (Memory Buffer Register) | MAR (Memory Address Register)
PC (Program Counter)
IR (Instruction Register)
AC (Accumulator)
ALU (Arithmetic Logic Unit)

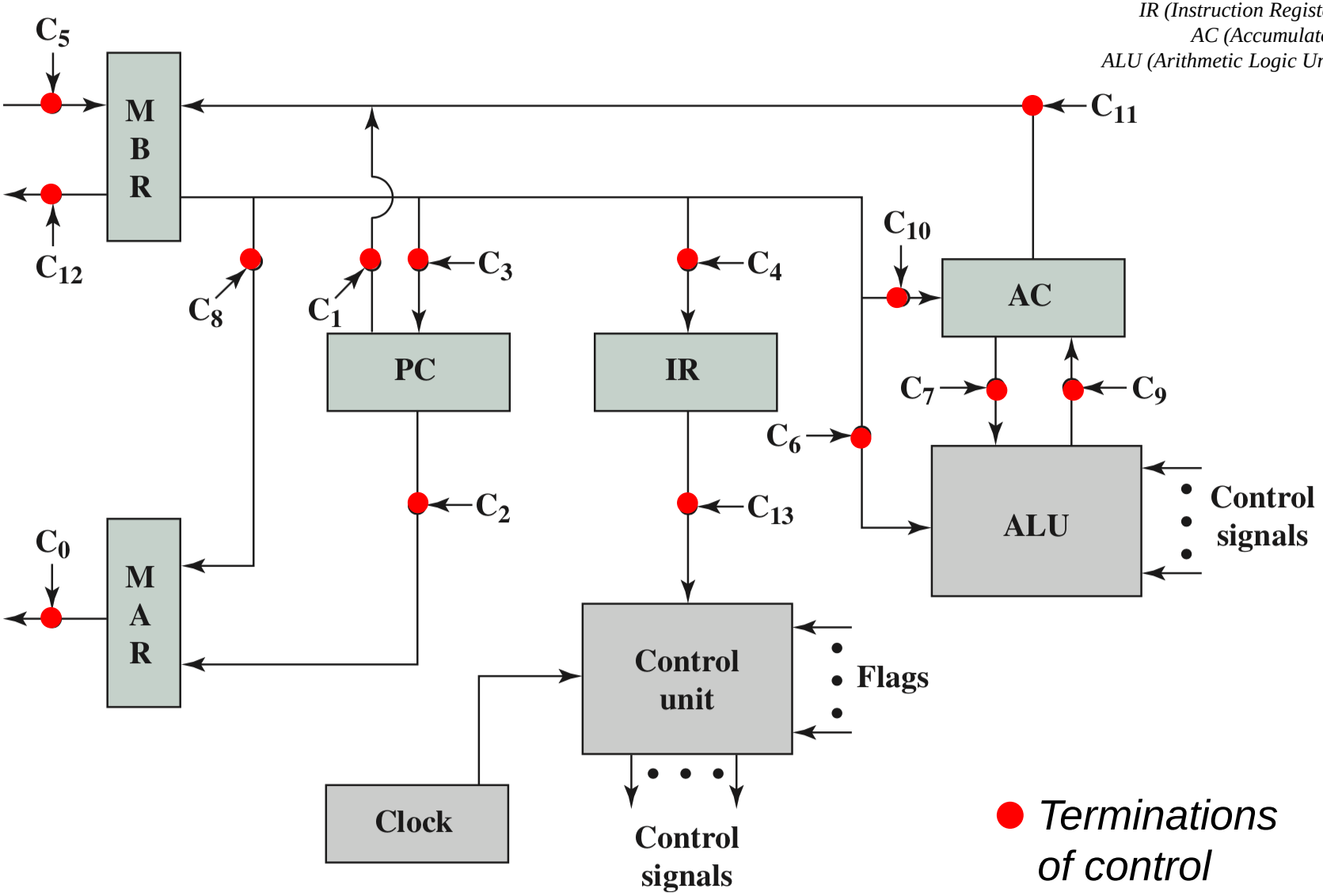


Figure: Data paths and control signals.

	Micro-operations	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

C_R = Read control signal to system bus.

C_W = Write control signal to system bus.

Figure: Micro-operation and control signals.

Module 5c
Central Processing Unit (CPU)

5.1 Processor Organization
5.2 Register Organization
5.3 Instruction Cycles
5.4 Instruction Pipelining
5.5 Control Unit Operation
5.6 Microprogrammed Control
5.7 Summary

- Overview
- Micro-Operations (2)
- Microprogrammed Control Unit
- Address Generation
- Microinstruction Format
- Advantages & Disadvantages of Microprogramming

5 Overview

- The term microprogram is an approach to Control Unit (CU) design that was organized systematically and avoided the complexities of a hardwired implementation.
- In recent years, microprogramming has become less used but remains a tool available to computer designers.
- **Example**: Pentium 4 → most of instructions are executed without the use of microprogramming, but some of the instructions are executed using microprogramming.

- A wide variety of techniques have been used for CU. Most of these fall into one of two categories:
  - (a) Hardwired implementation
  - (b) Microprogrammed implementation

(a) Hardwired Implementation

- Sequential circuits.
- The control logic is implemented with gates, flip-flops, decoders, and other digital circuits.

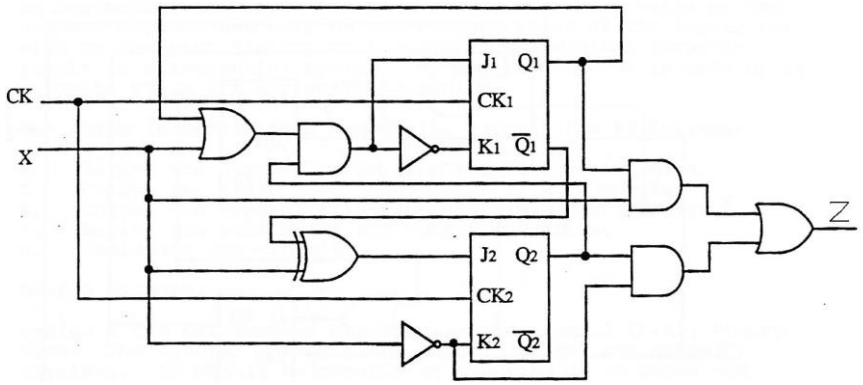


Figure: Example of sequential circuit.

Fast operation, small (requires less components).

Needs wiring change (if the design has to be modified)

Figure resource: https://web.sonoma.edu/users/m/marivani/es210/units/images/fig10_1.png (3 Apr 2019)

(b) Microprogrammed Implementation

5

- The control information is stored in a control memory.
- the control memory is programmed to initiate the required sequence of micro-operations (μOP).

Systematic, and any required change can be done by updating the micro-program in control memory.

Slow operation, requires more components.

5 Micro-Operations

- To implement a Control Unit (CU) as an interconnection of basic logic elements is no easy task.
- The design must include logic for sequencing through micro-operations, for executing micro-operations, for interpreting opcodes, and for making decisions based on ALU flags.

- It is difficult to design and test such a piece of hardware.
- The design is relatively inflexible. For example, it is difficult to change the design if one wishes to add a new machine instruction.

Disadvantages of hardware implementation.

5

Solution:

- An alternative to a hardwired control unit is a microprogrammed control unit, in which the logic of the CU is specified by a microprogram (*firmware*).

- A microprogram consists of a sequence of instructions (*micro-operations*) in a microprogramming language.

- A microprogrammed control unit is a relatively simple logic circuit that is capable of :

  (1) sequencing through *micro-operations* and

  (2) generating control signals to execute each *micro-operation*.

How can we use the concept of microprogramming to implement a Control Unit (CU)?

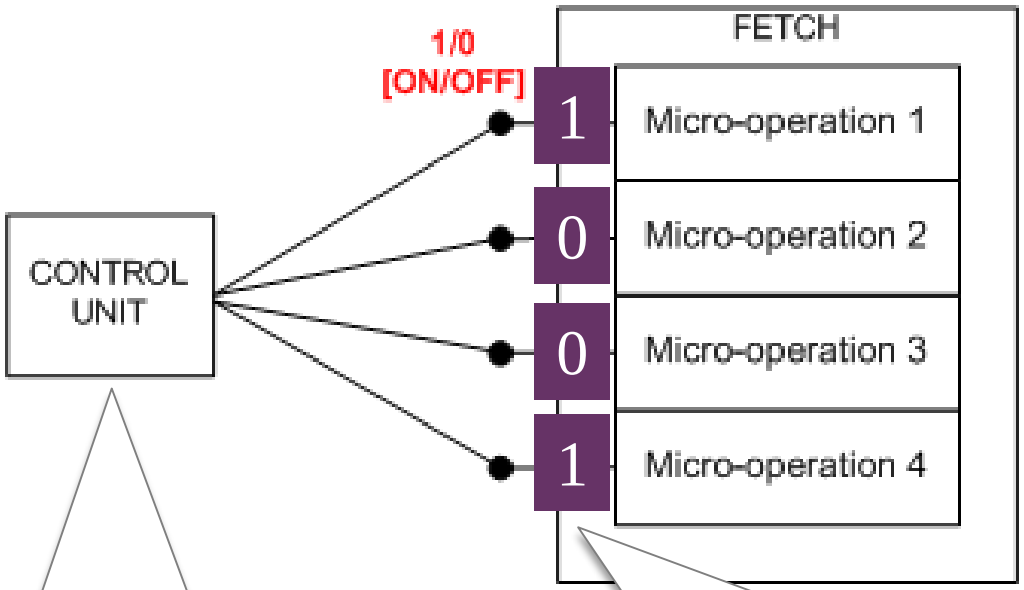
5

- Consider that for each micro-operation, the CU is allowed to generate a set of control signals.
- Thus, for any micro-operation, each control line emanating from the CU is either *on* (*1*) or *off* (*0*).
- So we could construct a control word (*microinstruction*) in which each bit represents one control line.

- Then each micro-operation would be represented by a different pattern of *1*s and *0*s in the microinstruction.

Example 20:

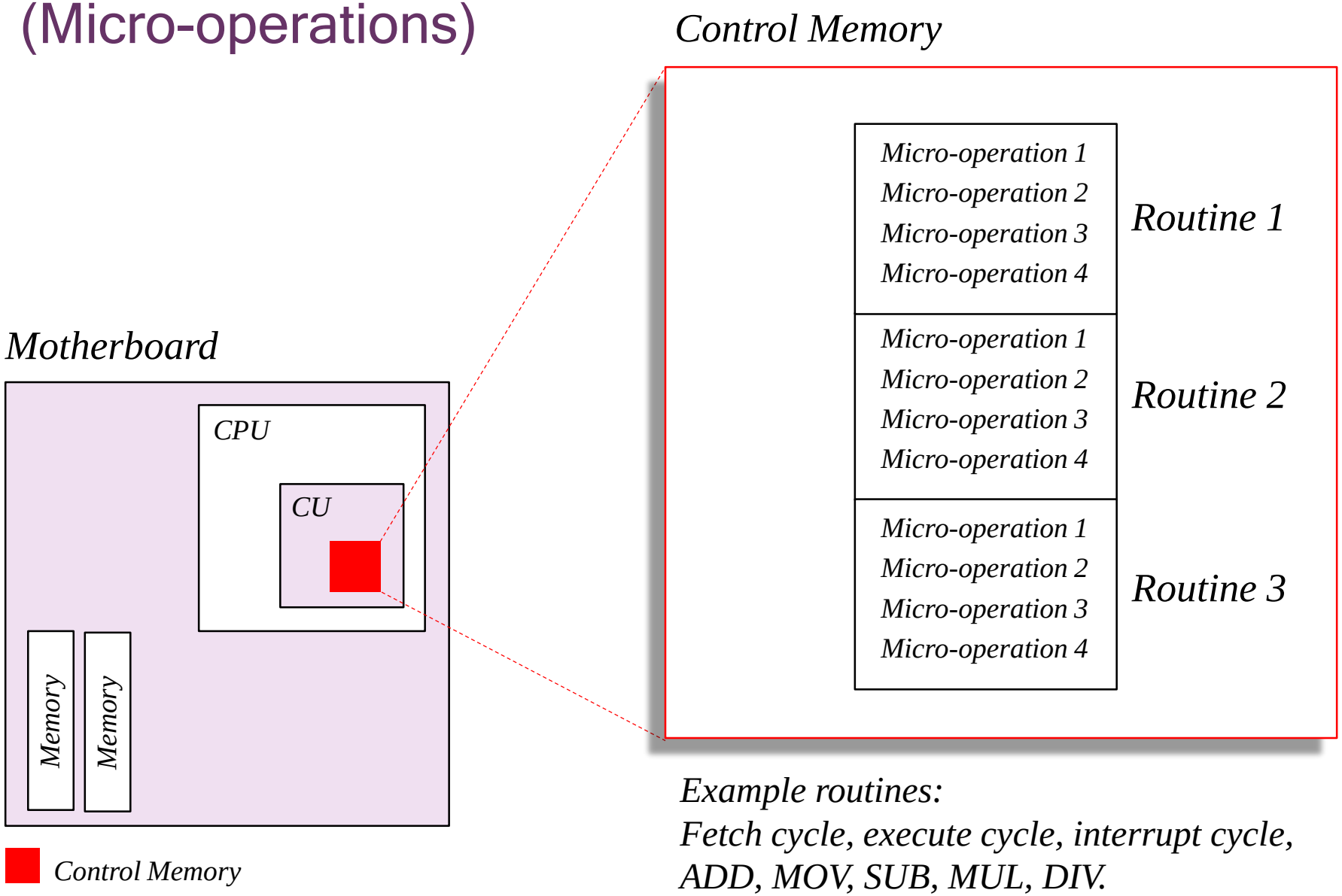
For *fetch instruction*, there are many micro-operations.



Control Unit sends on (1) or off (0) signal to the micro-operation

1010, 0011, 1100, 1110 *will initiate the different micro-operation to do the fetch instruction.*

(Micro-operations)



Microprogrammed Control Unit

(*Simple description*)

- The set of *micro-operations* is stored in the control memory.
- The control address register contains the address of the next *micro-operation* to be read.
- When a *micro-operation* is read from the control memory, it is transferred to a control buffer register .

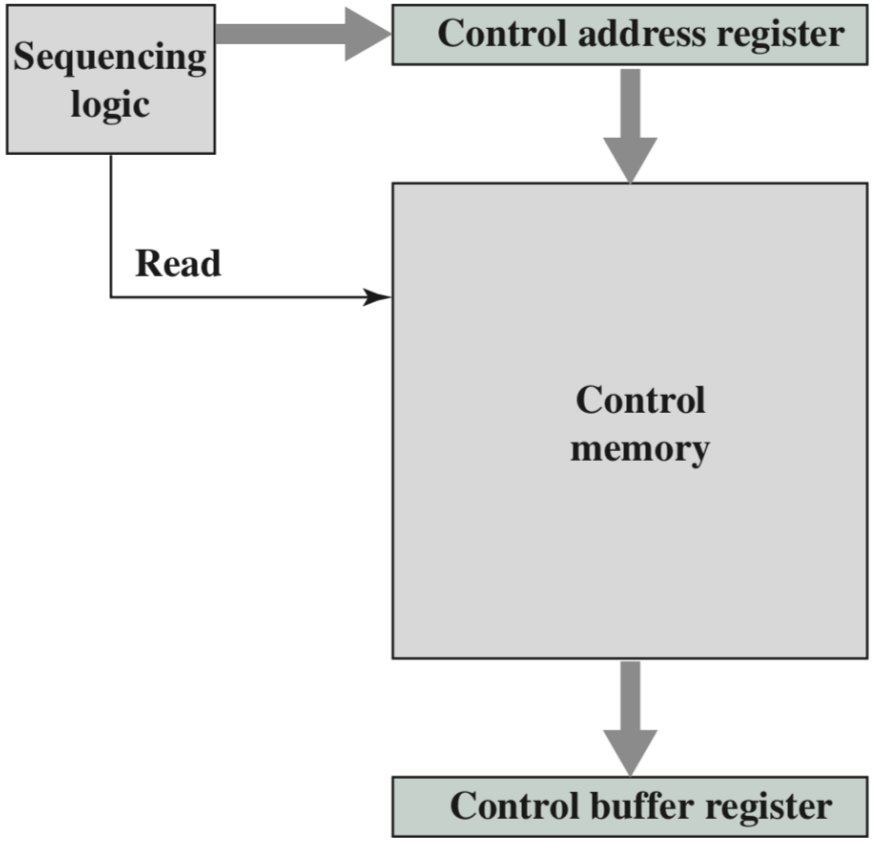
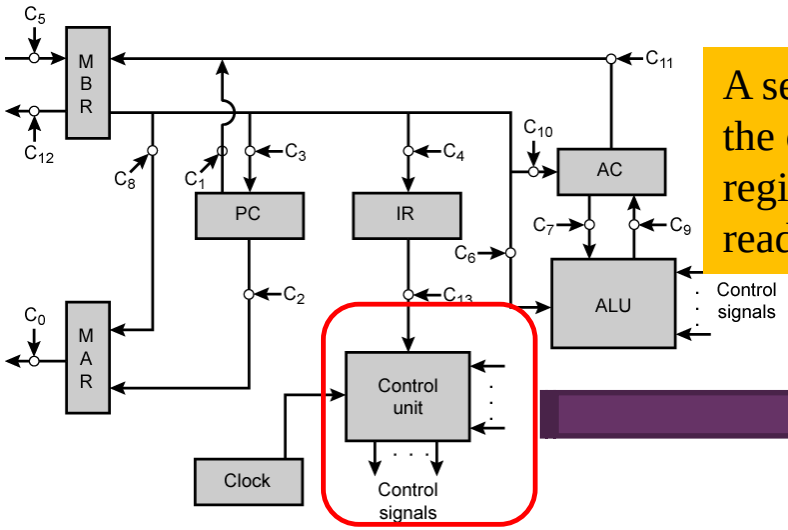


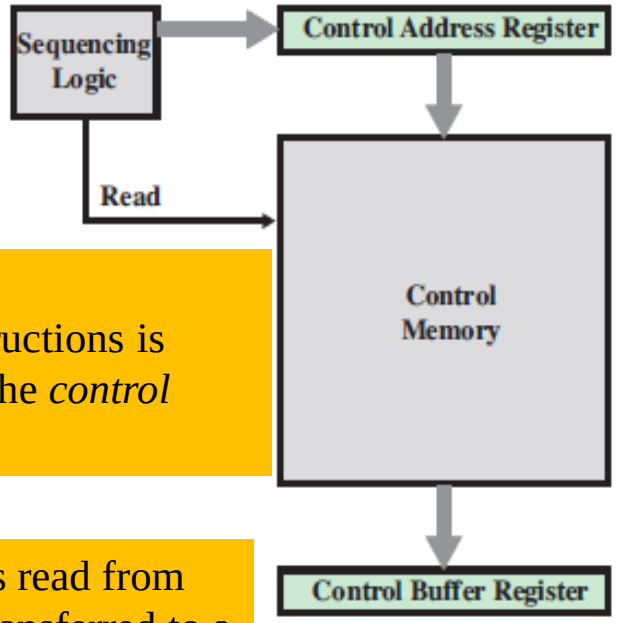
Figure: Control Unit Microarchitecture

Control Unit Microarchitecture



A sequencing unit loads the control address register and issues a read command.

The *control address register contains the address of the next microinstruction to be read.*



The set of microinstructions is stored in the *control memory.*

When a microinstruction is read from the control memory, it is transferred to a *control buffer register.*

- Figure shows the programs (*routines*) could be arranged in a control memory.

The micro-operations in each routine are to be executed sequentially.

Each routine ends with a branch or jump instruction indicating where to go next.

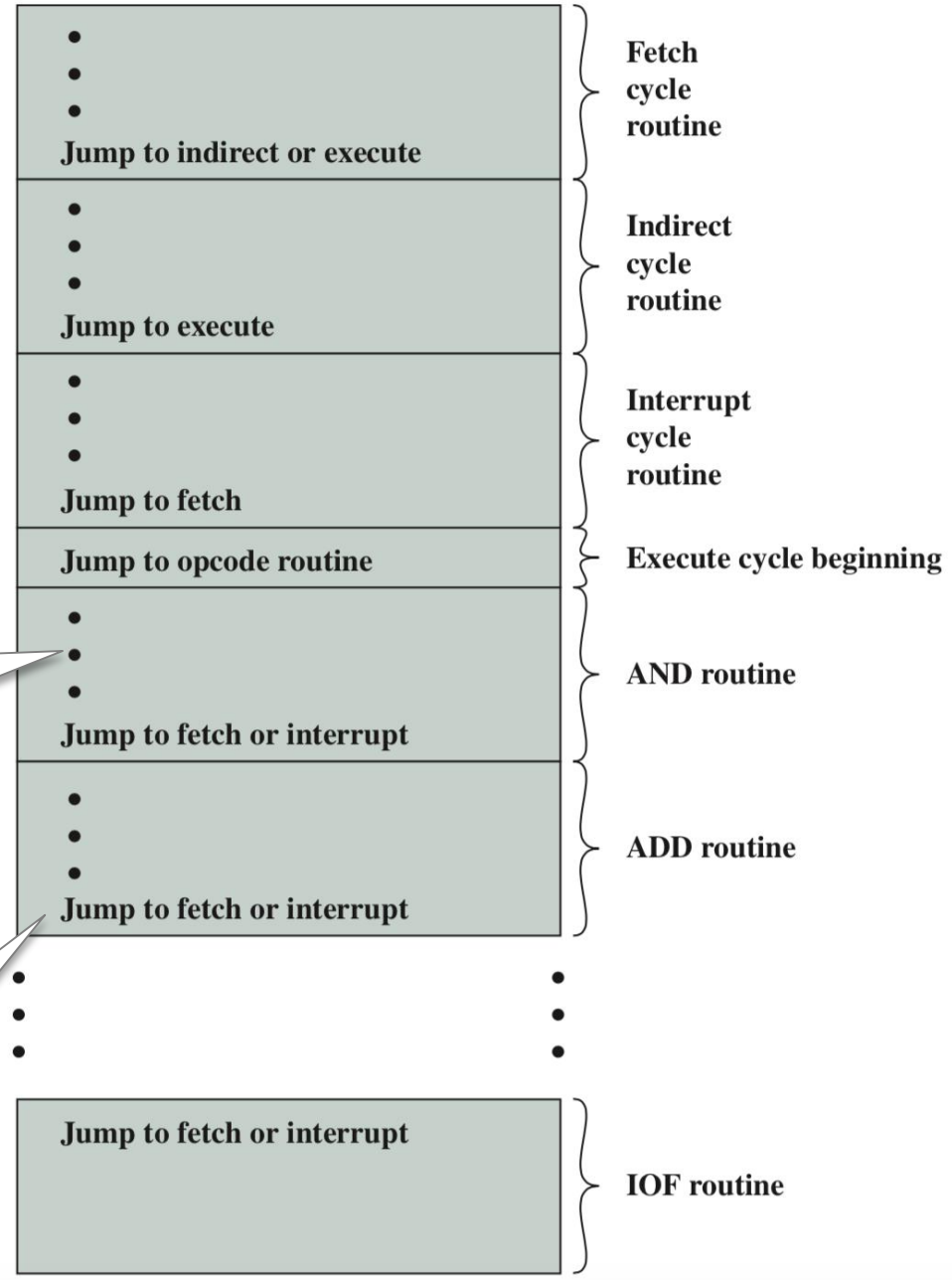


Figure: Organization of control memory

(*Detail description*)

- Comparing this figure with the previous figure (*simple description*), we see that the control unit still has the same inputs (IR, ALU flags, clock) and outputs (control signals).

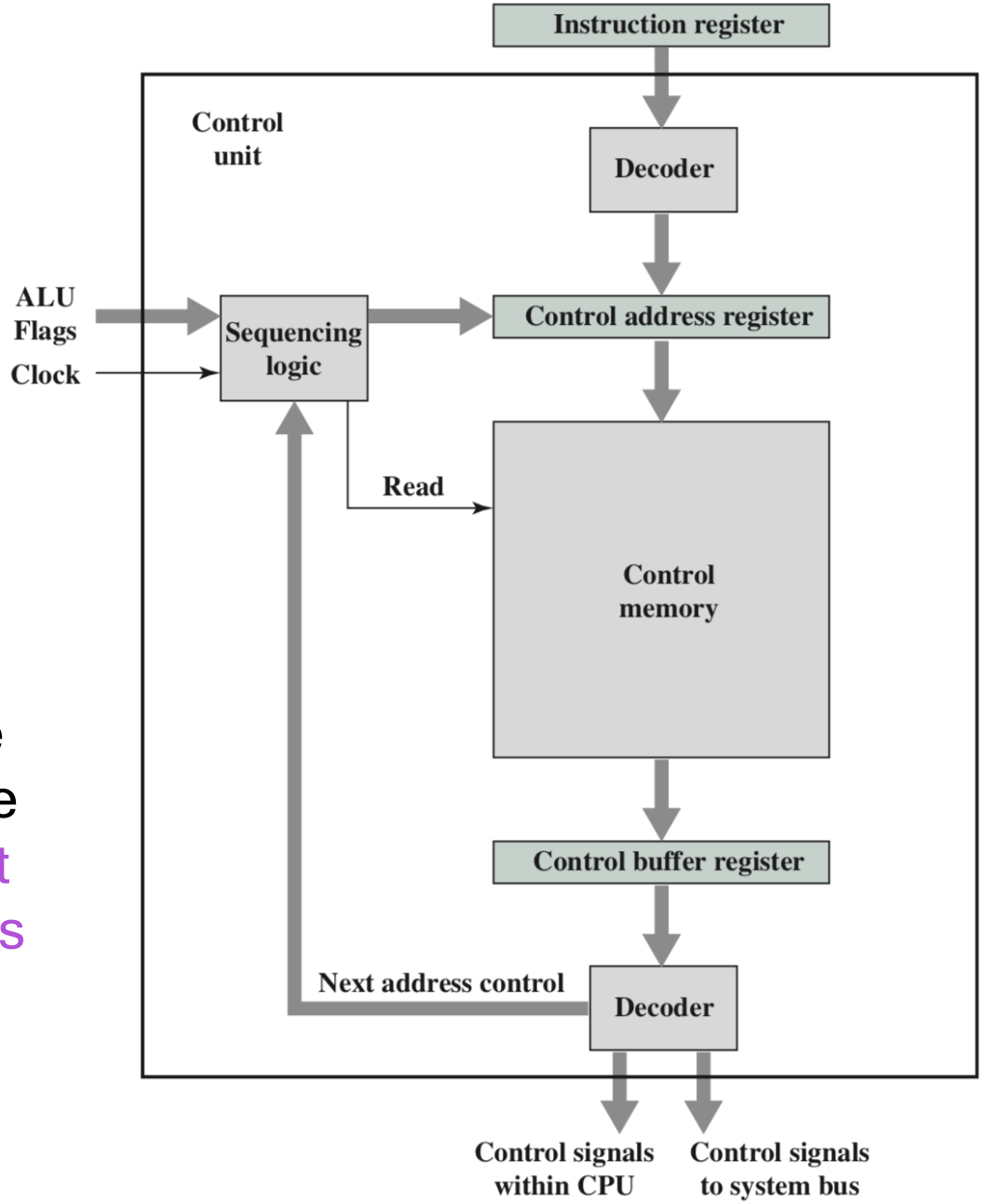


Figure: Functioning of Microprogrammed Control Unit

(1) To execute an instruction, the sequencing logic unit issues a READ command to the control memory.

(2) The word whose address is specified in the control address register is read into the control buffer register.

(3) The content of the control buffer register generates control signals and next-address information for the sequencing logic unit.

(4) The sequencing logic unit loads a new address into the control address register based on the next-address information from the control buffer register and the ALU flags.

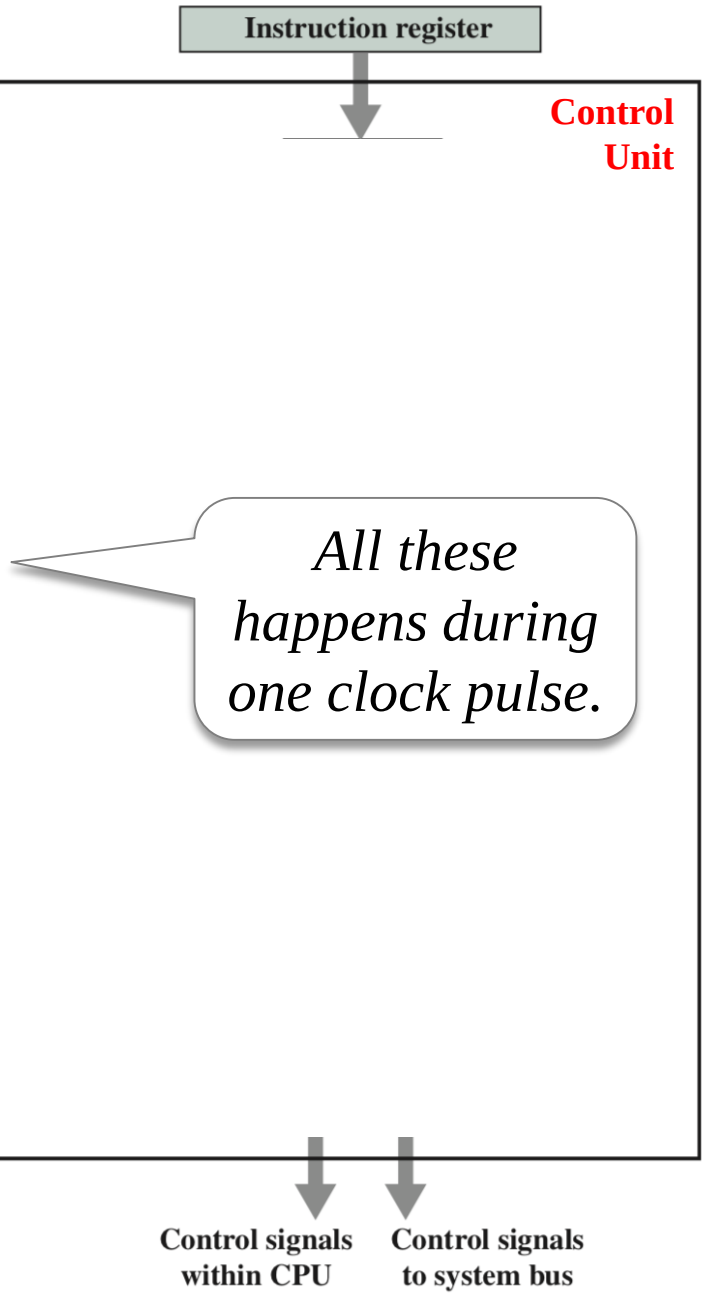
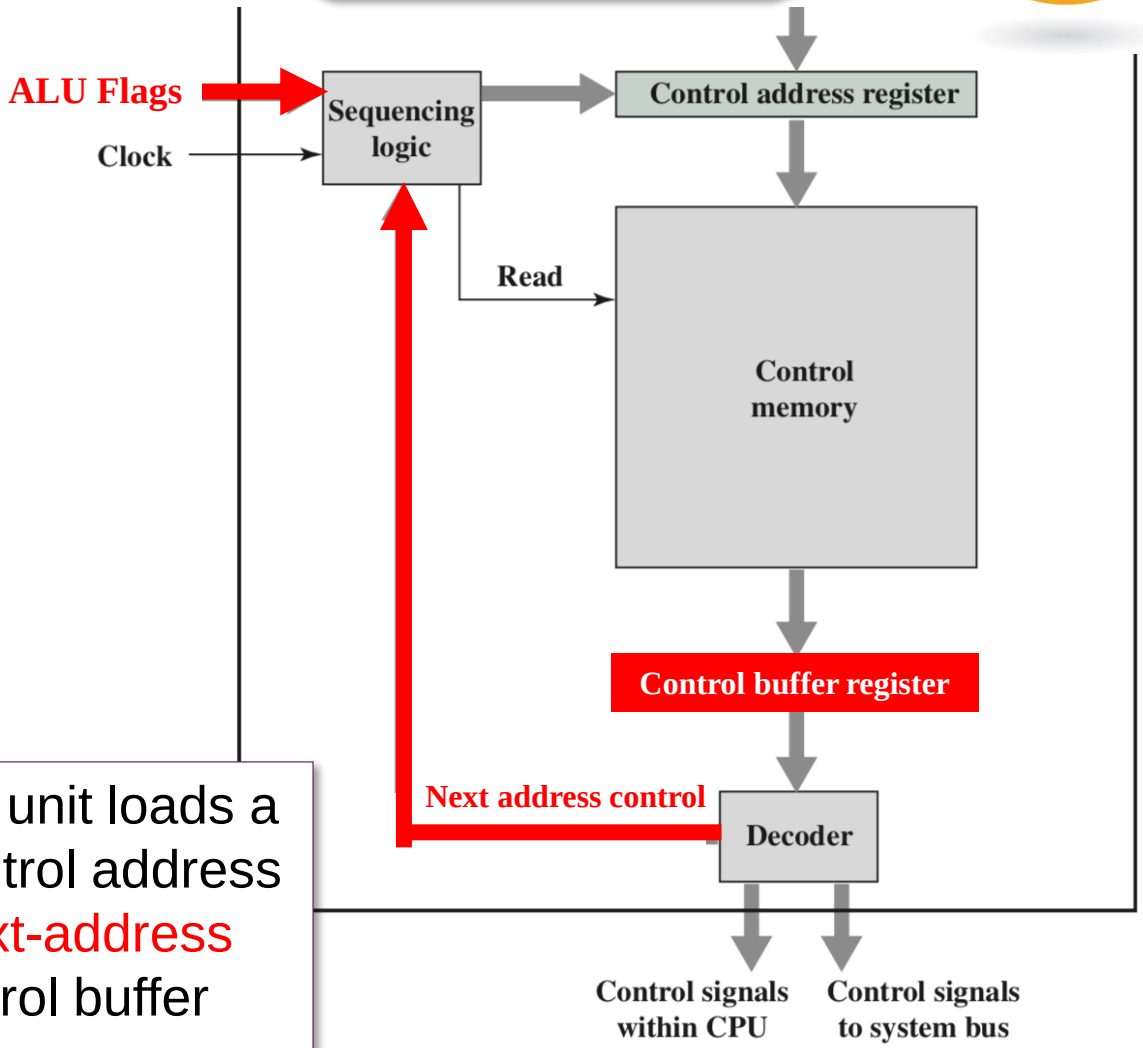


Figure: Functioning of Microprogrammed Control Unit

The last step just listed needs elaboration !



(4) The sequencing logic unit loads a new address into the control address register based on the next-address information from the control buffer register and the ALU flags.

The last step just listed needs elaboration !

(*Next address decision*)

- Depending on the value of the ALU flags and the control buffer register, one of three decisions is made:

(1) **Get the next instruction:** Add 1 to the control address register

(2) **Jump to a new routine based on a jump *micro-operation*:** Load the address field of the control buffer register into the control address register.

(3) **Jump to a machine instruction routine:** Load the control address register based on the opcode in the IR.

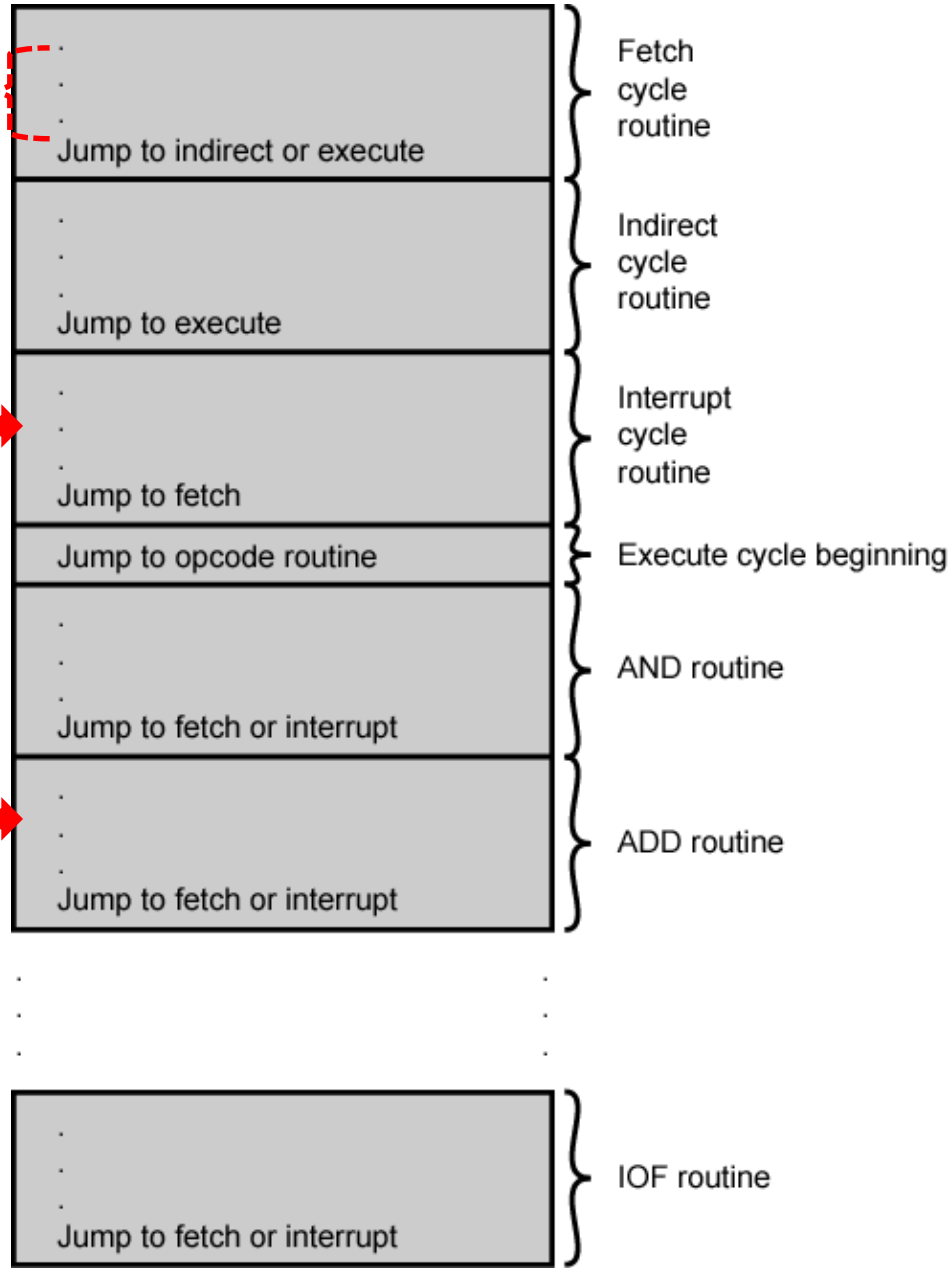
1. Get the next instruction: Add 1 to the control address register

2. Jump to a new routine based on a jump microinstruction:

Load the address field of the control buffer register into the control address register

3. Jump to a machine instruction routine:

Load the control address register based on the opcode in the IR.



Control Memory

5 Address Generation

- Another viewpoint is to consider the various ways in which the next address can be derived or computed.

Table: Microinstruction Address Generation Techniques

Explicit	Implicit
Two-field	Mapping
Unconditional branch	Addition
Conditional branch	Residual control

The address is explicitly available in the microinstruction.

Require additional logic to generate the address.

5

- *Microinstructions are stored in control memory in groups, which each group specifying a *routine*.
- *Each computer instruction has its own microprogram routine to generate *micro-operations* that execute the instruction.

*Mapping	The transformation from the instruction code bits (*opcode*) to an address in control memory where the routine (*micro-operation*) is located.

Mapping is one of several implicit techniques that commonly used !

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.743.
* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.216.

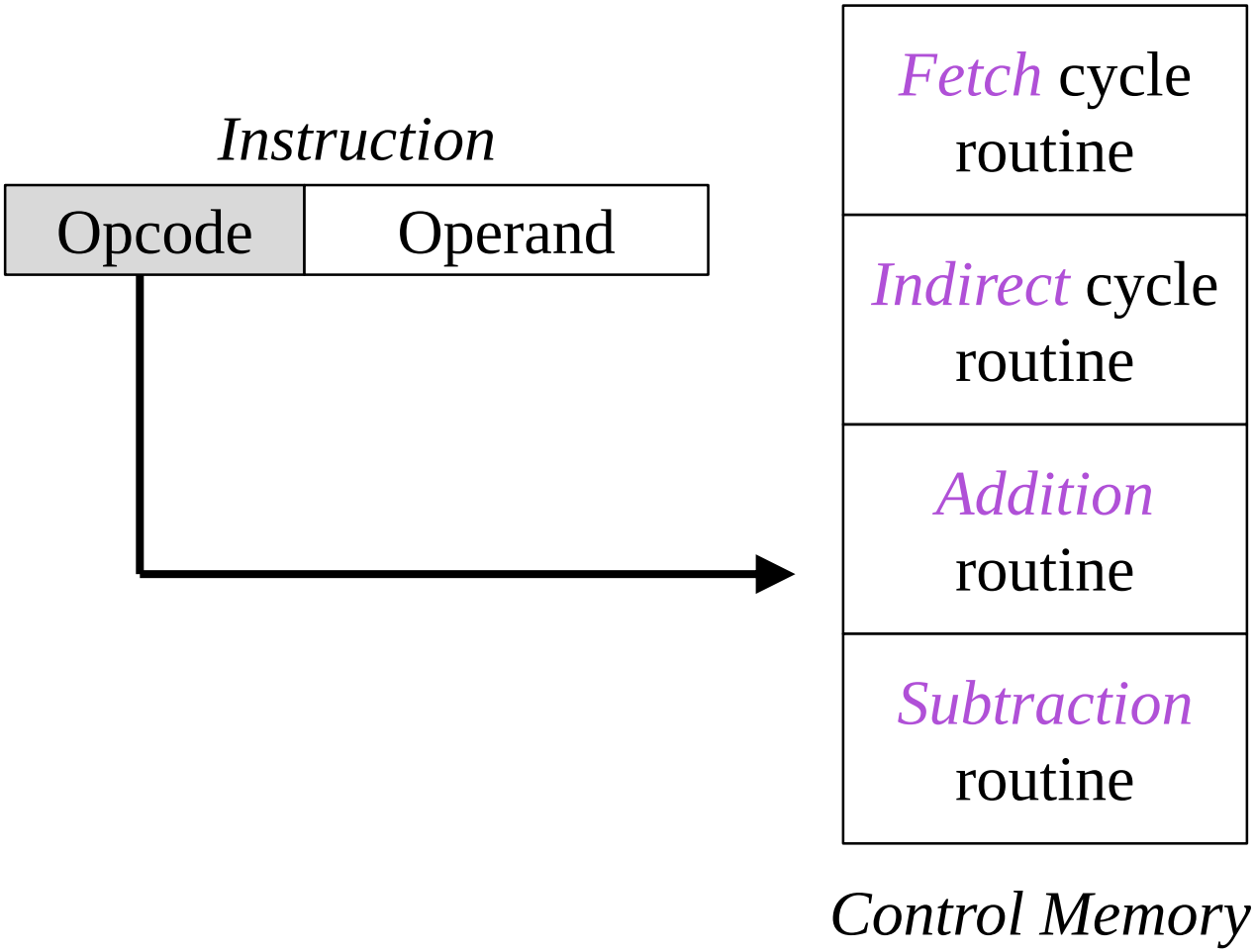


Figure: Transformation of opcode to microinstruction

Mapping of Instruction

- A special type of branch exist when a microinstruction specifies a branch to the first word in control memory where a microprogram routine for an instruction is located.

- Two methods:

Direct Mapping	Using ROM to specify mapping function

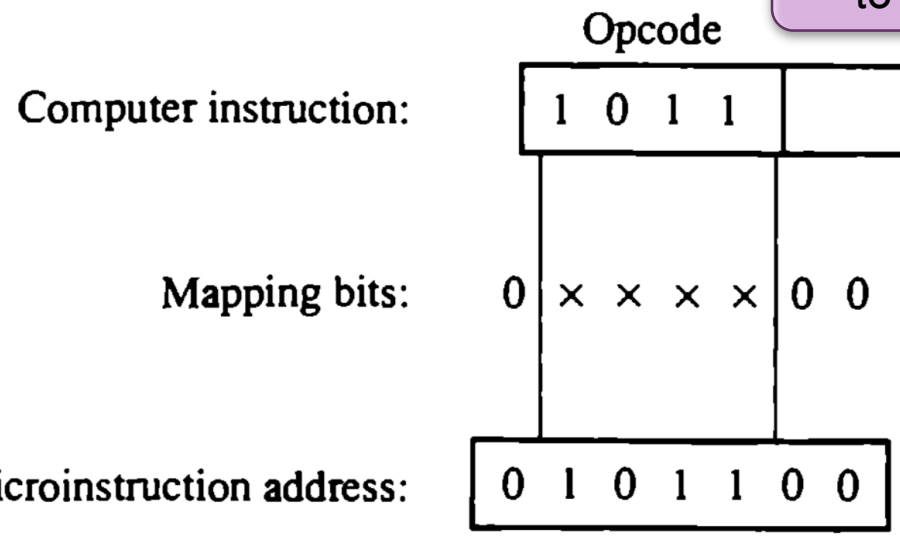
* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.219.

For each opcode, there exists a microprogram routine in control memory that execute the instruction. One simple mapping process that converts the 4-bit opcode to a 7-bit address for control memory

(a) Direct Mapping

- The status bits for this type of branch are the bits in the operation code part of the instruction.

4 bits opcode can specify up to 16 distinct instructions



Assume control memory has 128 words, requiring an address of 7 bits

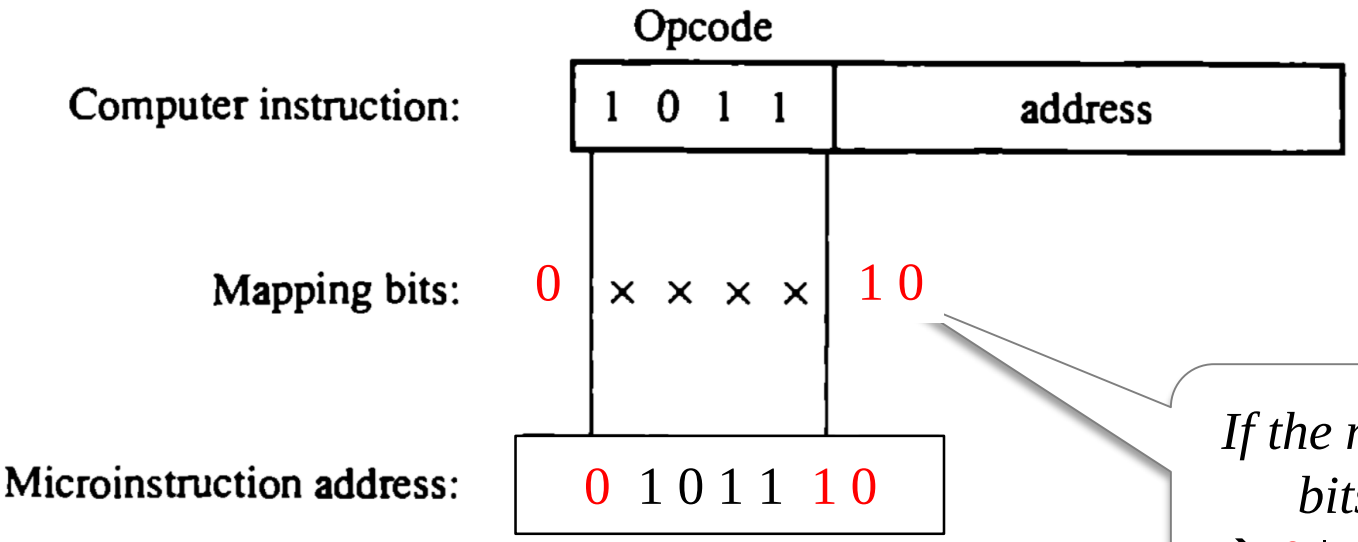
Figure: Mapping from instruction code to microinstruction address

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.219.

The mapping consists of placing a 0 in MSB of the address, transferring the four opcode bits and placing the two LSB of the control address register

Example 21:

Mapping from instruction code to microinstruction address.



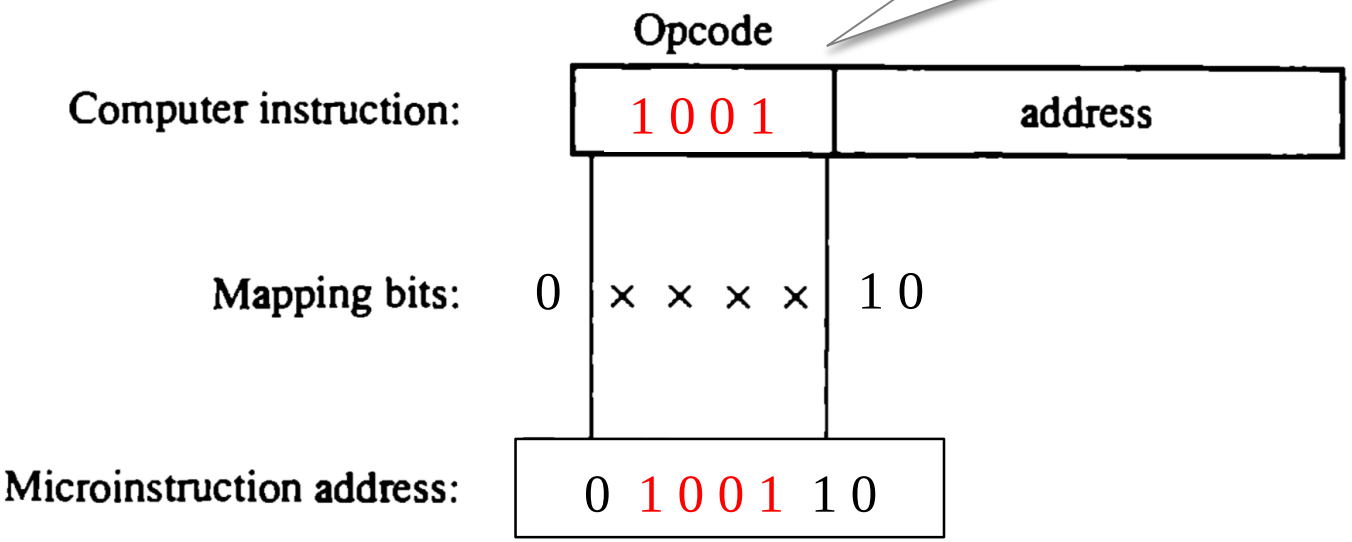
* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.219.

The mapping consists of placing a 0 in MSB of the address, transferring the four opcode bits and placing the two LSB of the control address register

Example 22:

Mapping from instruction code to microinstruction address.

If the opcode bits are → 1001



* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.219.

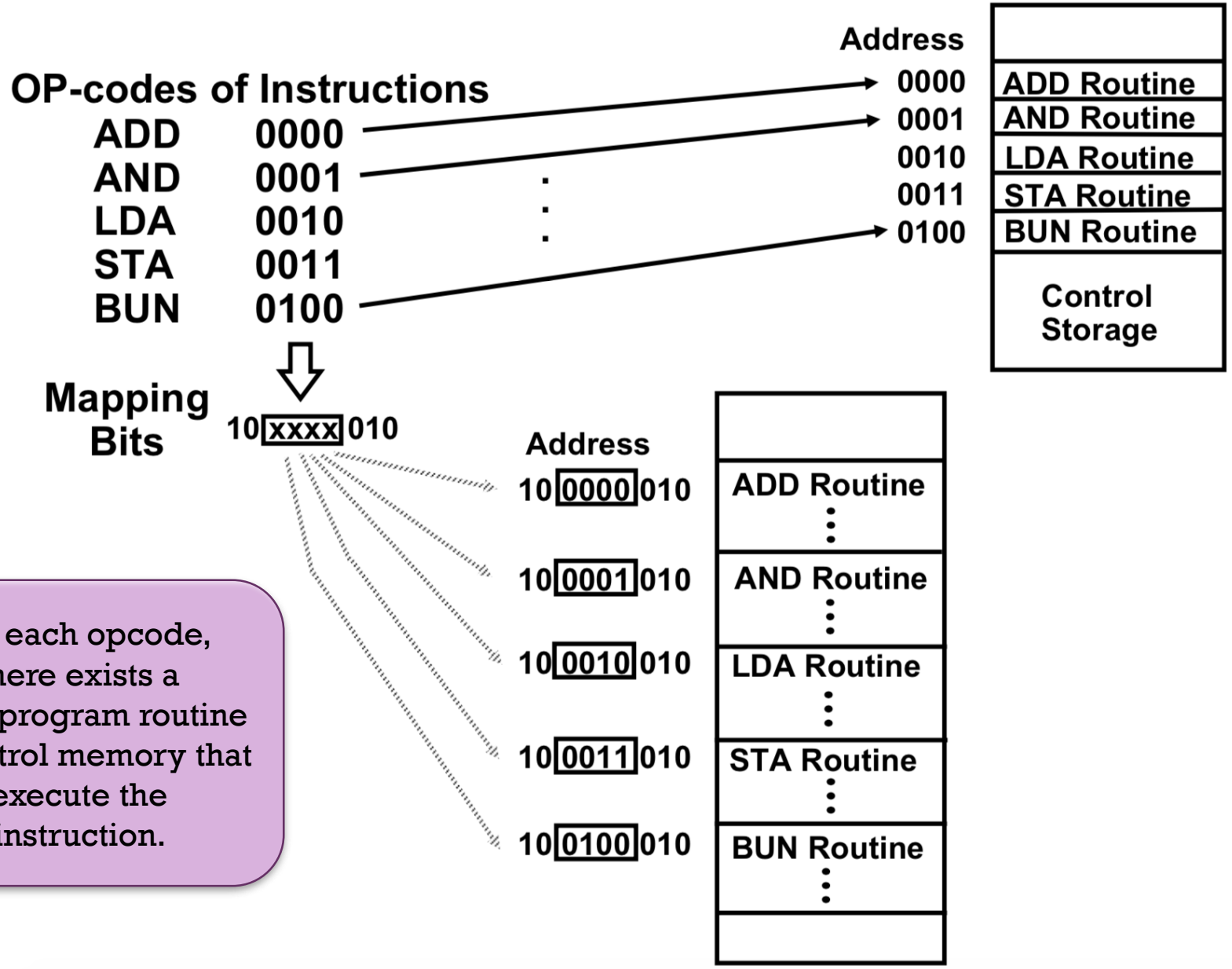
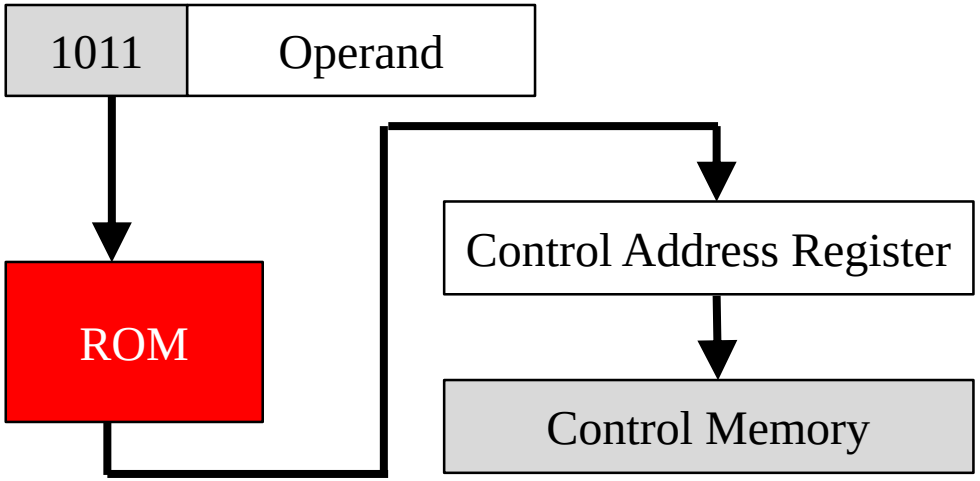


Figure: Direct mapping from instruction code to microinstruction address

(b) Using ROM to Specify Mapping Function

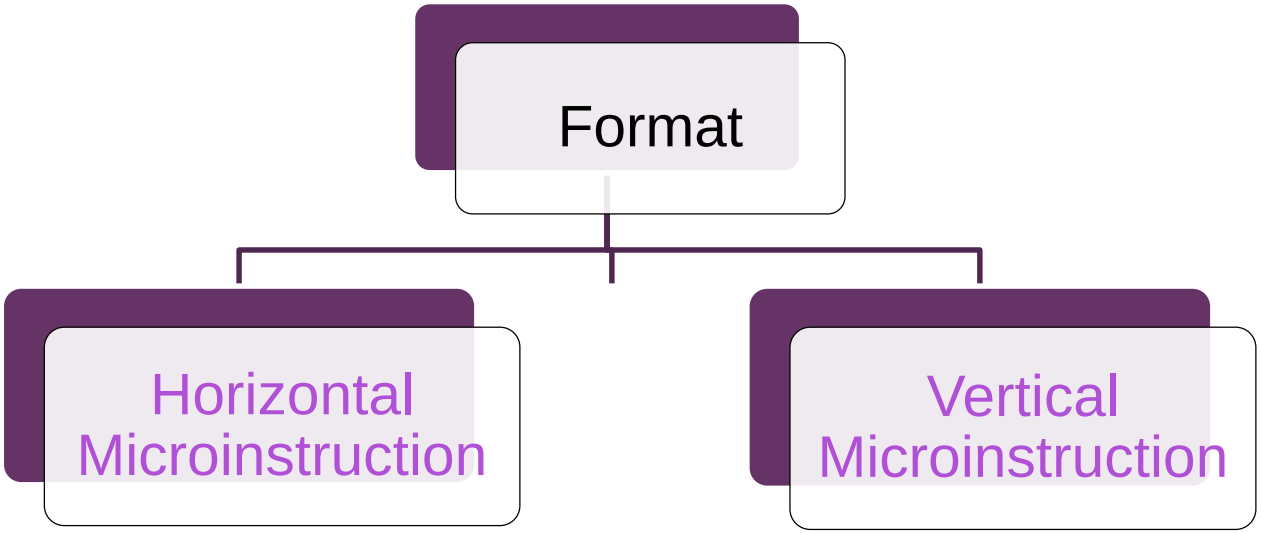
- One can extend the concept of direct mapping with more general rules by using a ROM to specify the mapping function.
- The bits of the instruction specify the address of a mapping ROM.
- The contents of the mapping ROM gives the bits for the control address register.



*The microprogram routine can be anywhere in the **control memory** → provides flexibility for adding instructions for **control memory** as need arise.*

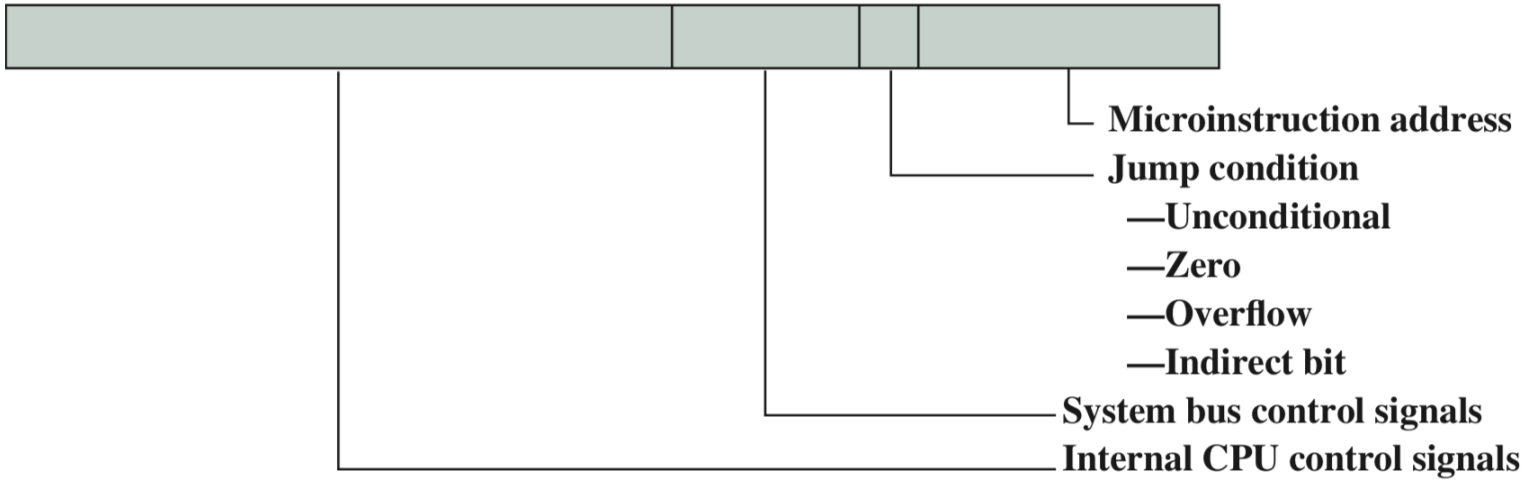
* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.219.

5 Microinstruction Format

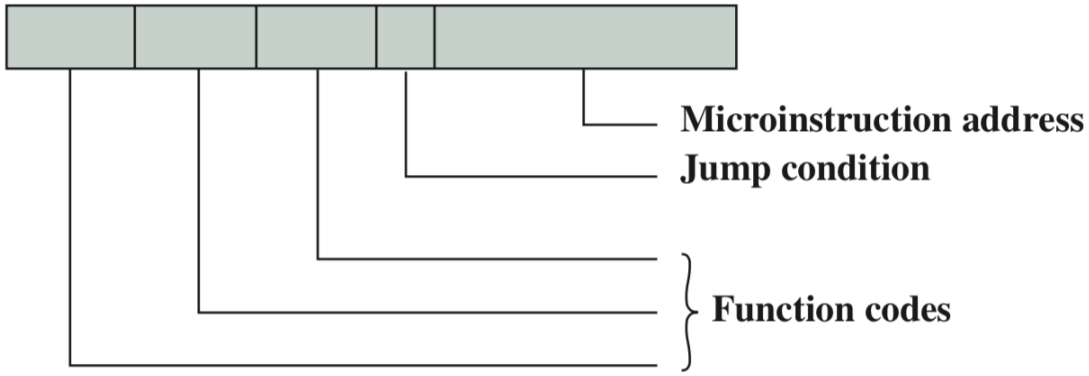


Each microinstruction specifies many different micro-operations to be performed in parallel.

Each microinstruction specifies single (or few) micro-operations to be performed.



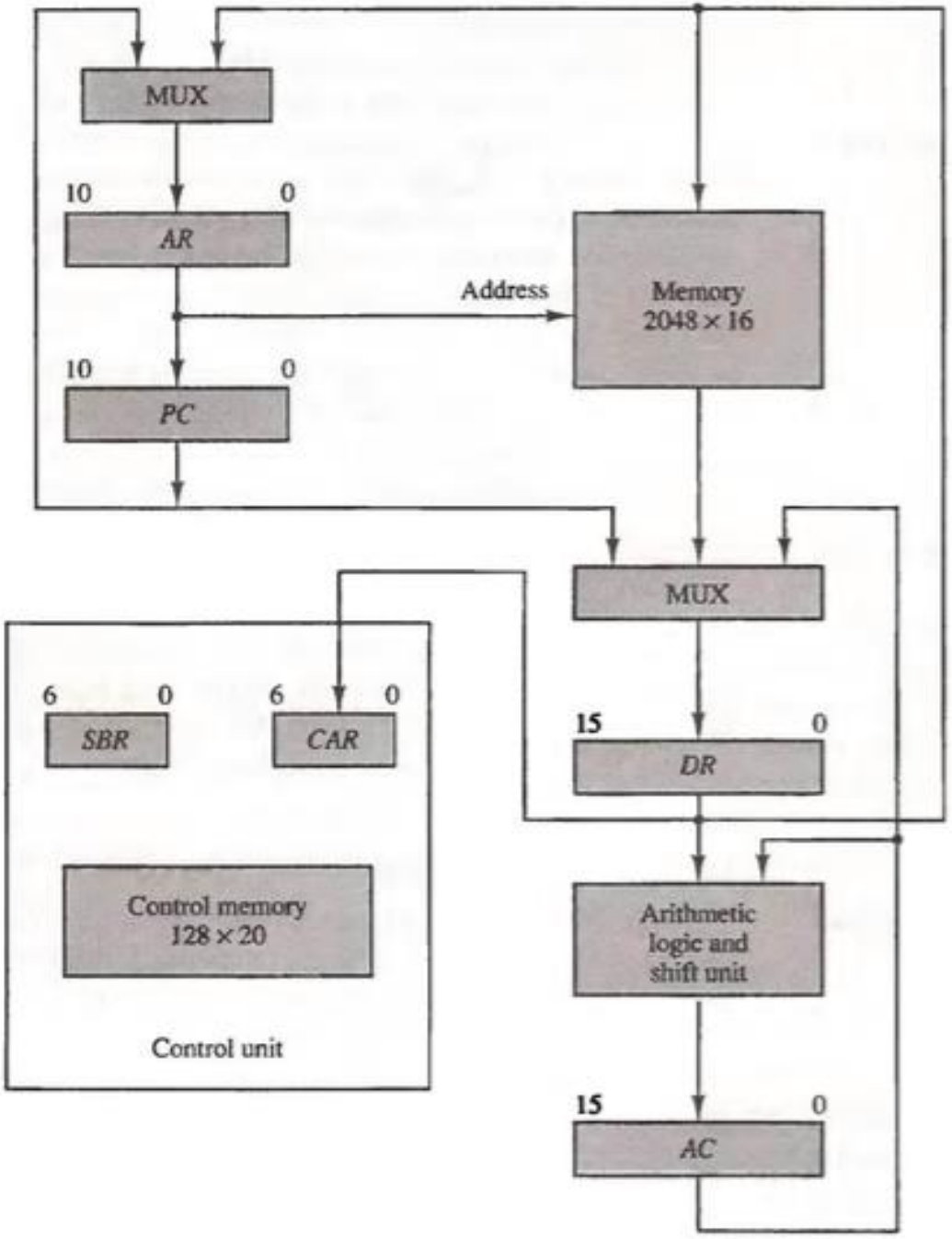
(a) Horizontal microinstruction



(b) Vertical microinstruction

Figure: Typical Microinstruction formats

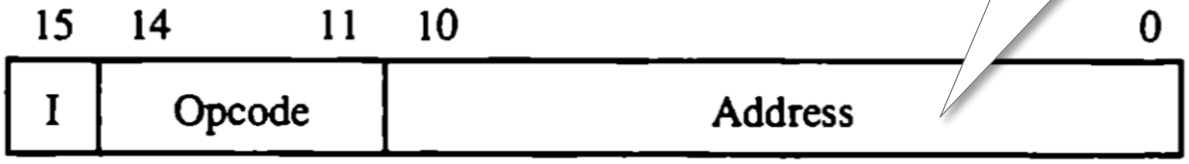
Figure: Computer hardware configuration



* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.221.

I → Indirect Addressing

11 bits address field



(a) Instruction format

4 bits opcode can have 16 possible instructions

Symbol	OP-code	Description
ADD	0000	AC ← AC + M[EA]
BRANCH	0001	if (AC < 0) then (PC ← EA)
STORE	0010	M[EA] ← AC
EXCHANGE	0011	AC ← M[EA], M[EA] ← AC

EA (Effective Address)

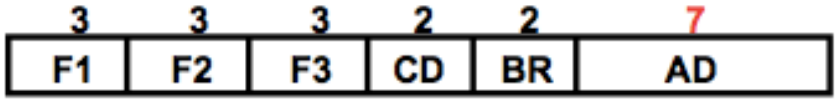
Figure: Sample of four computer instructions

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.222.

5

Vertical microinstruction format

Microinstruction Format



F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.223.

F1	Micro-operation	Symbol
000	None	NOP
001	AC ← AC + DR	ADD
010	AC ← 0	CLRAC
011	AC ← AC + 1	INCAC
100	AC ← DR	DRTAC
101	AR ← DR(0-10)	DRTAR
110	AR ← PC	PCTAR
111	M[AR] ← DR	WRITE

F2	Micro-operation	Symbol
000	None	NOP
001	AC ← AC - DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

F3	Micro-operation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← AC'	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

Microinstruction field description:

F1, F2, F3

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.224.

The condition for branching

Microinstruction field description:

CD, BR

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	Sign bit of AC
11	AC = 0	Z	Zero value in AC

The type of branch to be used

BR	Symbol	Function
00	JMP	CAR ← AD if condition = 1
		CAR ← CAR + 1 if condition = 0
01	CALL	CAR ← AD, SBR ← CAR + 1 if condition = 1
		CAR ← CAR + 1 if condition = 0
10	RET	CAR ← SBR (Return from subroutine)
11	MAP	CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.224.

F2	Microoperation	Symbol
000	None	NOP
001	AC ← AC - DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← AC'	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

Example 23:

A microinstruction with 2 simultaneous micro-operations.

DR ← M [AR] ----------→ {F2 = 100}

PC ← PC + 1 ----------→ {F3 = 101}

Since no F1 micro-operation, {F1 = 000}

Thus the vertical microinstruction fields:

F1	F2	F3			
000	100	101	CD	BR	AD

Let's recall;
The two micro-operations are from t2 of fetch cycle

MBR (Memory Buffer Register) | MAR (Memory Address Register)
PC (Program Counter)
IR (Instruction Register)

	Micro-operations	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

C_R = Read control signal to system bus.

C_W = Write control signal to system bus.

Figure: Micro-operation and control signals.

No more than 3 micro-operations can be chosen for a single microinstruction.

F1:
F2: DR ← M [AR]
F3: PC ← PC + 1

F1	Microoperation	Symbol
000	None	NOP
001	AC ← AC + DR	ADD
010	AC ← 0	CLRAC
011	AC ← AC + 1	INCAC
100	AC ← DR	DRTAC
101	AR ← DR(0-10)	DRTAR
110	AR ← PC	PCTAR
111	M[AR] ← DR	WRITE

F2	Microoperation	Symbol
000	None	NOP
001	AC ← AC - DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← AC'	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

Partial Symbolic Microprogram

Label	Microops	CD	BR	AD
	ORG 0			
ADD:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE:	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 12			
EXCHANGE:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 64			
FETCH:	PCTAR	U	JMP	NEXT
	READ, INCPC	U	JMP	NEXT
	DRTAR	U	MAP	
INDRCT:	READ	U	JMP	NEXT
	DRTAR	U	RET	

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.228.

Micro Routine	Address		Binary Microinstruction					
	Decimal	Binary	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

This microprogram can be implemented using ROM

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.230.

Micro Routine	Address		Binary Microinstruction					
	Decimal	Binary	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
t_1: MAR ← (PC)	13	0001101	001	000	000	00	00	0001110
t_2: MBR ← Memory PC ← (PC) + 1	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
t_3: IR ← (MBR)								
FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

This microprogram can be implemented using ROM

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.230.

Example 24: Calculation example.

A system uses a control memory of 1024 words of 32 bits each. The microinstruction has 3 fields and a 16-bit micro-operations.

a) How many bits are there in the branch address field and the selection field?

b) If there are 16 status in the system, how many bits of the branch logic are used to select a status bit?

c) How many bits are left to select an input for the multiplexers?

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.235.

Solution :

The keywords:

A system uses a control memory of 1024 words of 32 bits each.
The microinstruction has 3 fields and a 16-bit micro-operations

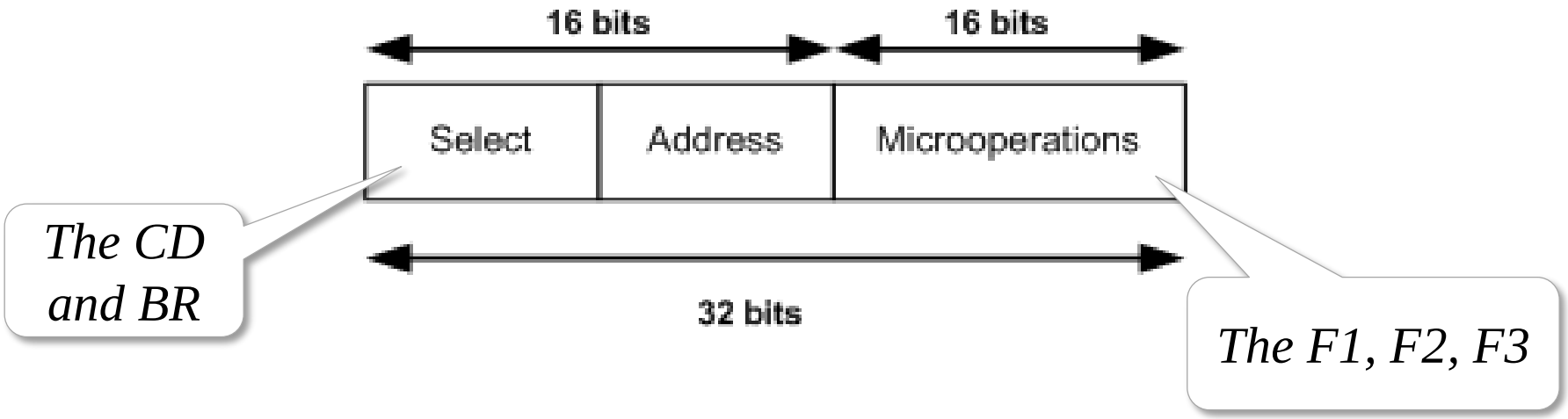


Figure: The microinstruction diagram

5

- The Control memory has 1024 words (2^{10}). Thus, the required address field for each word is 10 bits.
- The selection field consists of status bits and branch bits.
- The selection field will be: 32 – (16 + 10) = 6 bits

The micro-operations total bits

The bit for address field

5

a) How many bits are there in the branch address field and the selection field?

- The address field will be: 10 bit (2^{10} = 1024 words)
- The selection field will be: 32 – (16+10) = 6 bits

b) If there are 16 status in the system, how many bits of the branch logic are used to select a status bit?

- 16 status means 4 bits are required for it (2^4)

c) How many bits are left to select an input for the multiplexers?

- Input for the multiplexer = selection field bits – status bits
  = 6 – 4 = 2 bits

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.235.

Advantages & Disadvantages of Microprogramming

- It simplifies the design of the control unit.
- Thus, it is both cheaper and less error prone to implement.
- The decoders and sequencing logic unit are very simple pieces of logic.

- It will be somewhat slower than a hardwired unit of comparable technology

Despite this, microprogramming is the dominant technique for implementing CU in pure CISC architectures, due to its ease of implementation.

Summary 3

- Explained the concept of *micro-operations* and define the principal instruction cycle phases in terms of micro-operations.
- Discussed how micro-operations are organized to control a processor.
- Understand hardwired control unit organization.

Review Questions

5

5.1. Explain the steps in the fetch-decode-execute cycle. Your explanation should include what is happening in the various register.

5.2. Write an assembly language program to evaluate the expression. $A \times B + C \times D$

5.3. Write the following code segment in an assembly language:

(a)
```
X := 1;
while X < 10 do
   X := X + 1;
endwhile;
```

(b)
```
Sum := 0;
for X := 1 to 10 do
   Sum := Sum + X;
```

5

5.4. What is the different between a microprocessor and microprogram? Is it possible to design a microprocessor without a microprogram? Are all microprogrammed computers also microprocessor?

5.5. Explain the different between hardwired and microprogrammed control. Is it possible to have a hardwired control associated with a control memory?

5.6. Define the following:

(a) Microoperation.

(b) Microinstruction.

(c) Microprogram.

(d) Microcode.

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.235.

5

5.6. Given the mapping bit from instruction code to microinstruction address as 0 xxxx 00. Give the microinstruction address for the following operation code.

(a) 0010 (b) 1011 (c) 1111

5.7. Using the table of symbol and binary code for microinstruction fields, give the 9-bit microoperation field for the following microoperations:

(a) $AC \leftarrow AC + 1, \quad DR \leftarrow DR + 1$

(b) $PC \leftarrow PC + 1, \quad DR \leftarrow M[AR]$

(c) $DR \leftarrow AC, \quad AC \leftarrow DR$

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.235-236; Table p.224

5

5.8. Using the table of symbol and binary code for microinstruction fields, convert the following symbolic microoperations to register transfer statement and to binary:

(a) $READ,\ \ INCPC$

(b) $ACTDR,\ \ DRTAC$

(c) $ARTPC,\ \ DRTAC,\ \ WRITE$

* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.236; Table p.224