به نام خدا

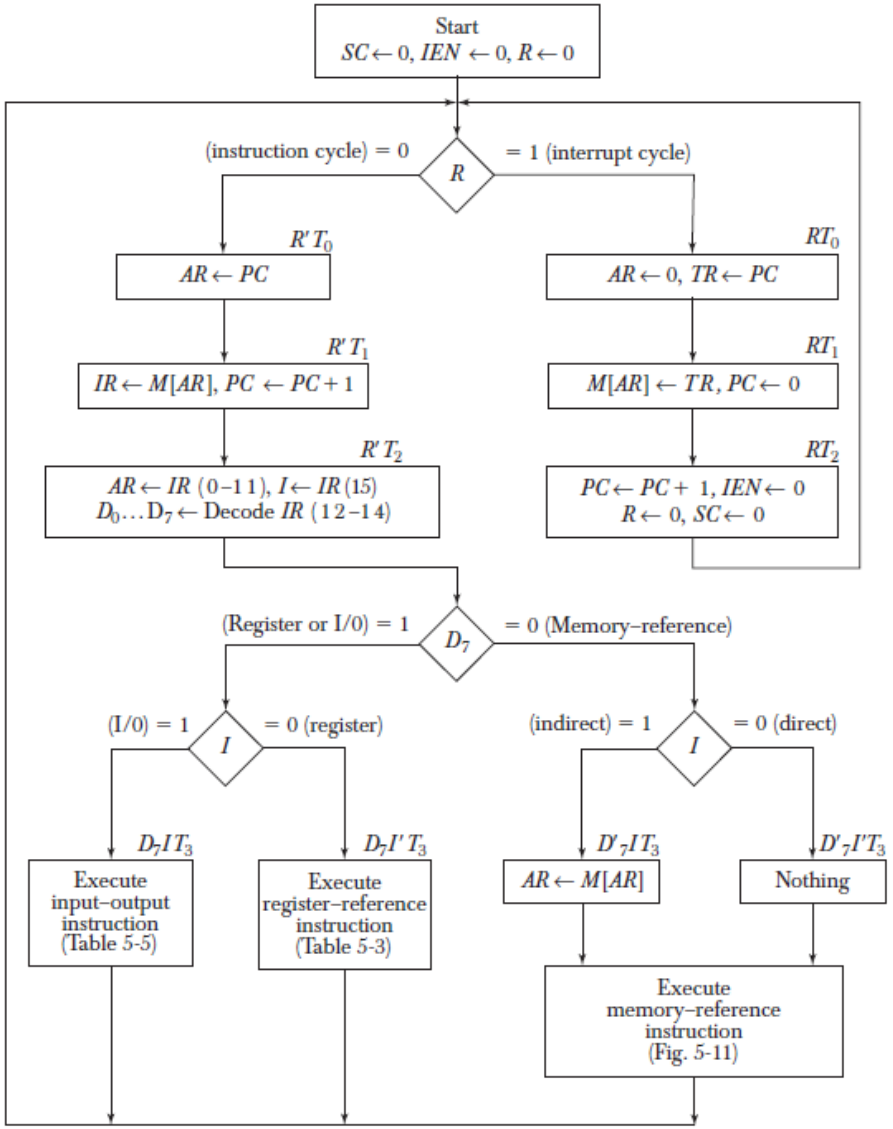
طراحی واحد کنترل

Microprogrammed

Dr. Aref Karimiafshar
A.karimiafshar@iut.ac.ir

مرور

- Instruction Cycle
  - Fetch
  - Decode
  - Read the effective address
  - Execute

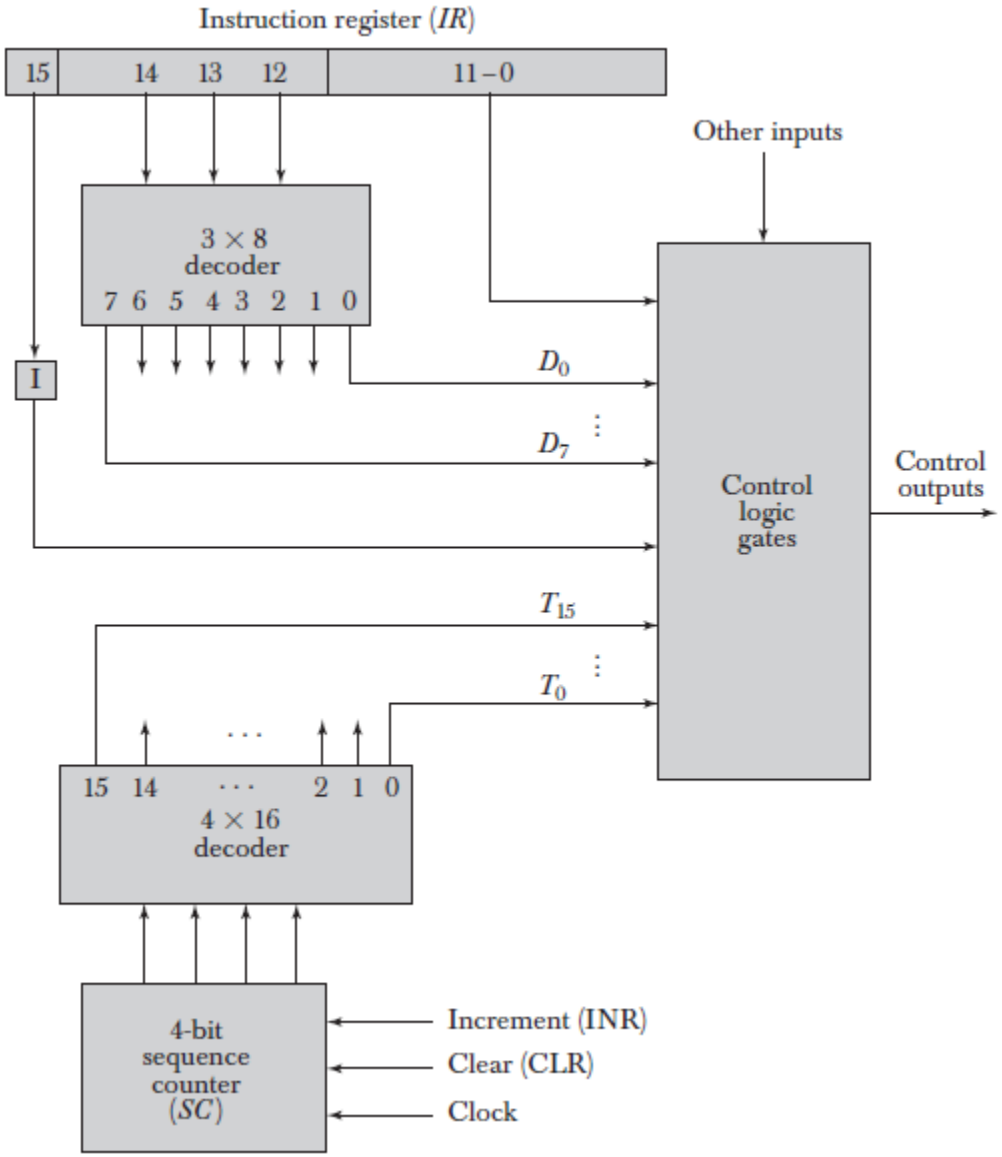


Control Functions and Microoperations

Control Functions and Microoperations for the Basic Computer

Fetch	$R'T_0$:	$AR \leftarrow PC$
	$R'T_1$:	$IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
Decode	$R'T_2$:	$D_0, \ldots, D_7 \leftarrow$ Decode $IR(12\text{–}14)$, $AR \leftarrow IR(0\text{–}11)$. $I \leftarrow IR(15)$
Indirect	$D_7'IT_3$:	$AR \leftarrow M[AR]$
Interrupt:		
$T_0T_1'T_2'(IEN)(FGI + FGO)$:		$R \leftarrow 1$
	RT_0:	$AR \leftarrow 0$, $TR \leftarrow PC$
	RT_1:	$M[AR] \leftarrow TR$, $PC \leftarrow 0$
	RT_2:	$PC \leftarrow PC + 1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$
Memory-reference:		
AND	D_0T_4:	$DR \leftarrow M[AR]$
	D_0T_5:	$AC \leftarrow AC \wedge DR$, $SC \rightarrow 0$
ADD	D_1T_4:	$DR \leftarrow M[AR]$
	D_1T_5:	$AC \leftarrow AC + DR$, $E \leftarrow C_{out} \rightarrow SC \leftarrow 0$
LDA	D_2T_4:	$DR \leftarrow M[AR]$
	D_2T_5:	$AC \leftarrow DR$, $SC \leftarrow 0$
STA	D_3T_4:	$M[AR] \leftarrow AC$, $SC \leftarrow 0$
BUN	D_4T_4:	$PC \leftarrow AR$, $SC \leftarrow 0$
BSA	D_5T_4:	$M[AR] \leftarrow PC$, $AR \leftarrow AR + 1$
	D_5T_5:	$PC \leftarrow AR$, $SC \leftarrow 0$
ISZ	D_6T_4:	$DR \leftarrow M[AR]$
	D_6T_5:	$DR \leftarrow DR + 1$
	D_6T_6:	$M[AR] \leftarrow DR$, if $(DR = 0)$ then $(PC \leftarrow PC + 1)$, $SC \leftarrow 0$
Register–reference:		
	$D_7I'T_3 = r$ (common to all register-reference instructions	
	$IR(i) = B_i$ $(i = 0, 1, 2, \ldots, 11)$	
	r:	$SC \leftarrow 0$
CLA	rB_{11}:	$AC \leftarrow 0$
CLE	rB_{10}:	$E \leftarrow 0$
CMA	rB_9:	$AC \leftarrow \overline{AC}$
CME	rB_8:	$E \leftarrow \overline{E}$
CIR	rB_7:	$AC \leftarrow$ shr AC, $AC(15) \leftarrow E$, $E \leftarrow AC(0)$
CIL	rB_6:	$AC \leftarrow$ shl AC, $AC(0) \leftarrow E$, $E \leftarrow AC(15)$
INC	rB_5:	$AC \leftarrow AC + 1$
SPA	rB_4:	If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$
SNA	rB_3:	If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$
SZA	rB_2:	If $(AC = 0)$ then $PC \leftarrow PC + 1)$
SZE	rB_1:	If $(E = 0)$ then $(PC \leftarrow PC + 1)$
HLT	rB_0:	$S \leftarrow 0$
Input–output:		
	$D_7IT_3 = p$ (common to all input–output instructions)	
	$IR(i) = B_i$ $(i = 6, 7, 8, 9, 10, 11)$	
	p:	$SC \leftarrow 0$
INP	pB_{11}:	$AC(0\text{–}7) \leftarrow INPR$, $FGI \leftarrow 0$
OUT	pB_{10}:	$OUTR \leftarrow AC(0\text{–}7)$, $FGO \leftarrow 0$
SKI	pB_9:	If $(FGI = 1)$ then $(PC \leftarrow PC + 1)$
SKO	pB_8:	If $(FGO = 1)$ then $(PC \leftarrow PC + 1)$
ION	pB_7:	$IEN \leftarrow 1$
IOF	pB_6:	$IEN \leftarrow 0$

مرور

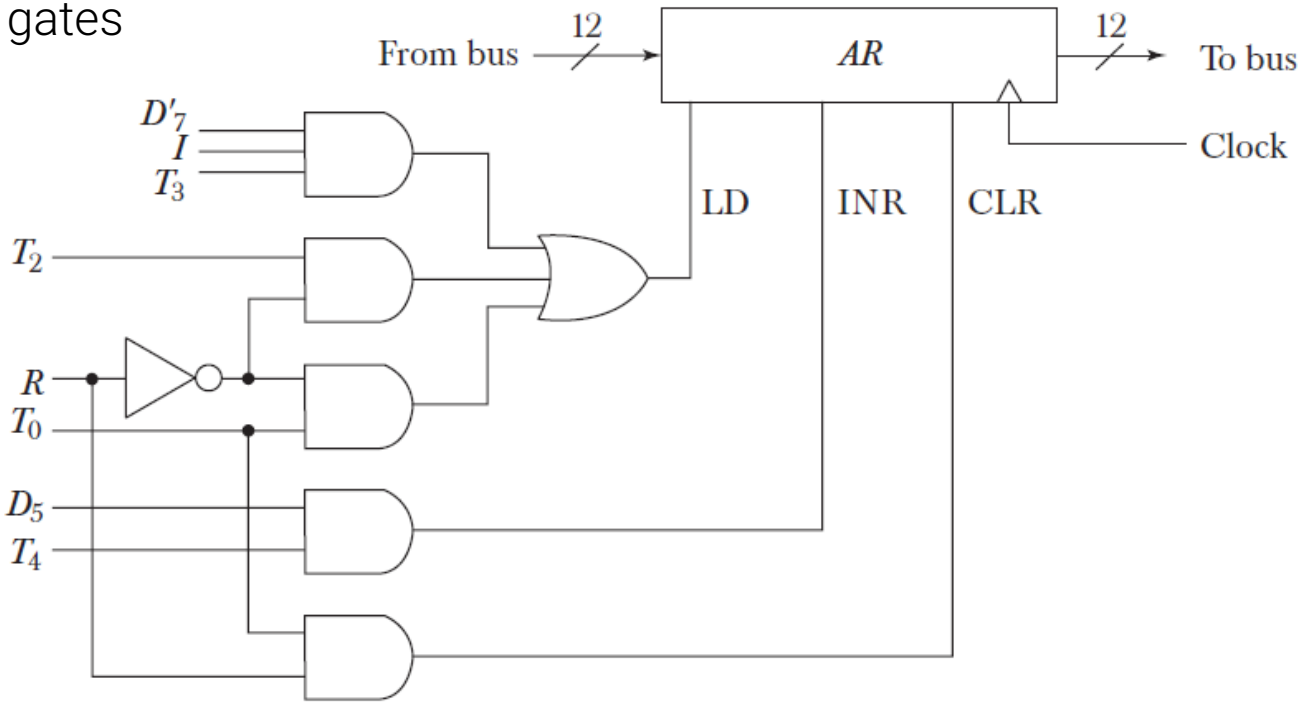

Instruction register (IR)
15
14
13
12
11 – 0
Other inputs
3 × 8
decoder
7 6 5 4 3 2 1 0
I
D_0
D_7
Control
logic
gates
Control
outputs
T_{15}
T_0
15 14 · · · 2 1 0
4 × 16
decoder
4-bit
sequence
counter
(SC)
Increment (INR)
Clear (CLR)
Clock

مرور

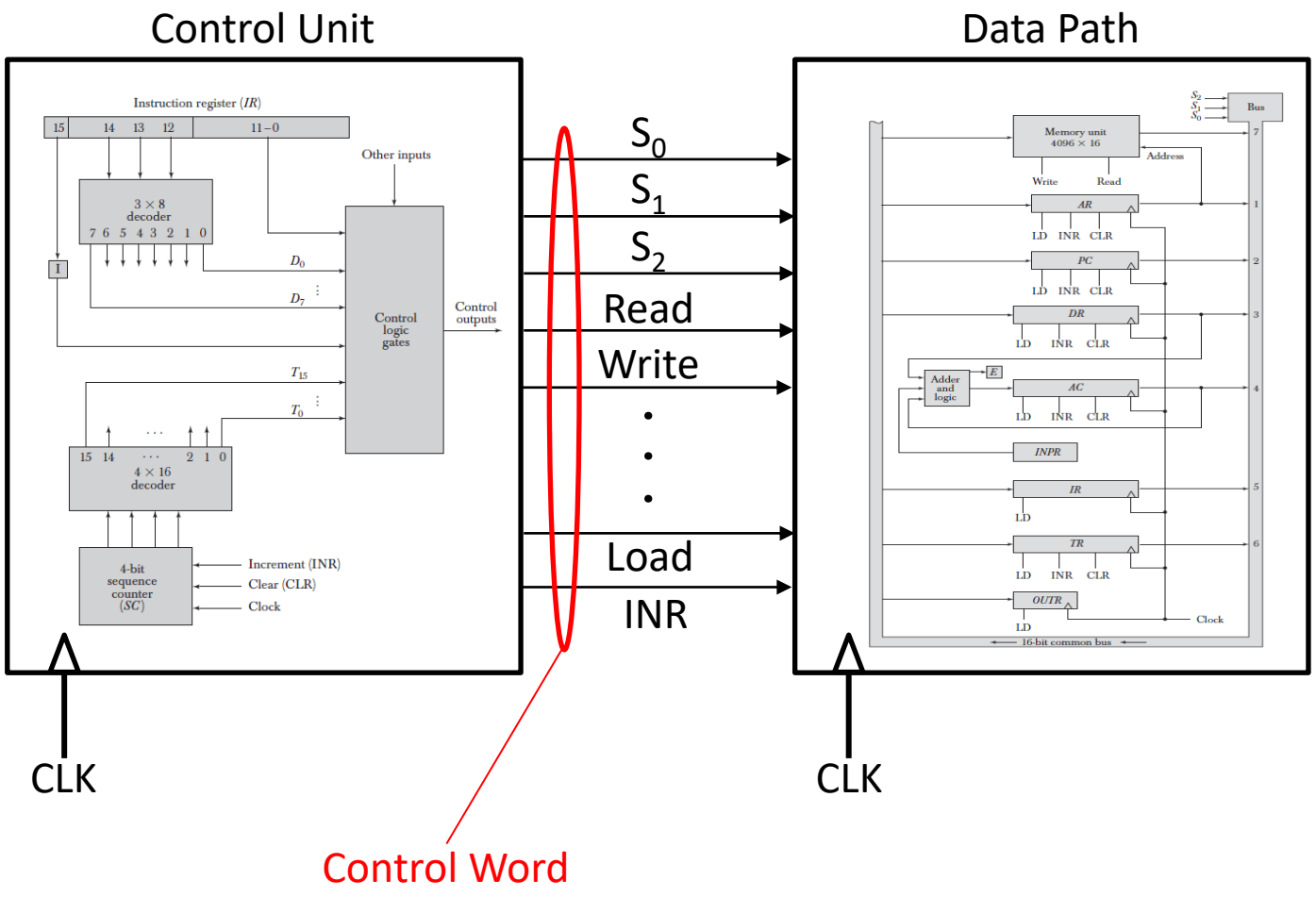
Control functions

$$\begin{aligned} \text{LD}(AR) &= R'T_0 + R'T_2 + D_7'IT_3 \\ \text{CLR}(AR) &= RT_0 \\ \text{INR}(AR) &= D_5T_4 \end{aligned}$$

Control gates



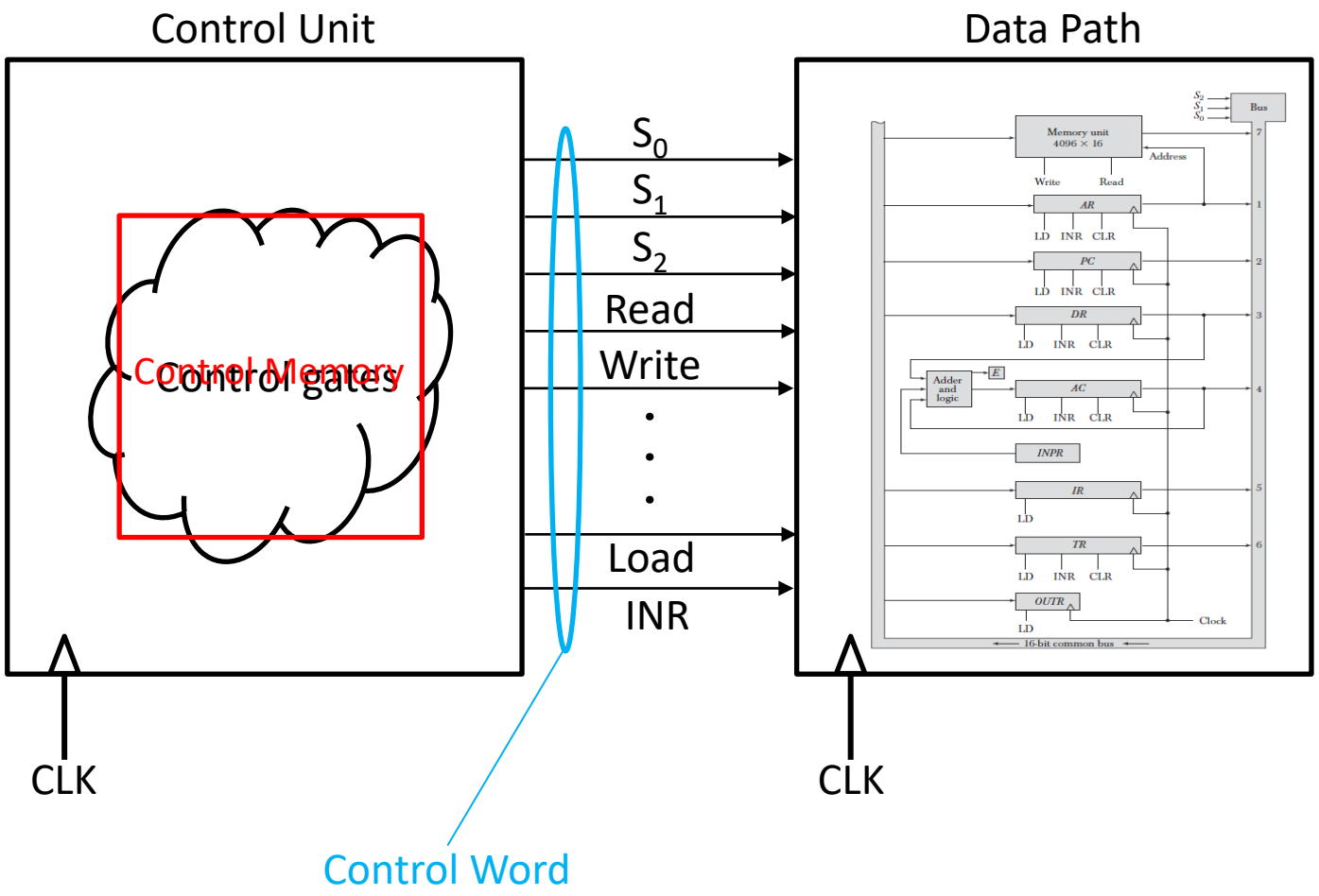
Control Unit



Control Word

- The control unit initiates a series of sequential steps of microoperations
  - During any given time, certain microoperations are to be initiated, while others remain idle
  - The control variables at any given time can be represented by a string of 1's and 0's called a control word
  - control words can be programmed to perform various operations on the components of the system
- A control unit whose binary control variables are stored in memory is called a microprogrammed control unit

Microprogrammed Control Unit



Microprogrammed Control Unit

- Each word in control memory contains within it a microinstruction
  - The microinstruction specifies one or more microoperations for the system
- A sequence of microinstructions constitutes a microprogram
- Since alterations of the microprogram are not needed once the control unit is in operation, the control memory can be a read-only memory (ROM)
  - The content of the words in ROM are fixed and cannot be altered by simple programming since no writing capability is available in the ROM

Microprogrammed Control Unit cnt.

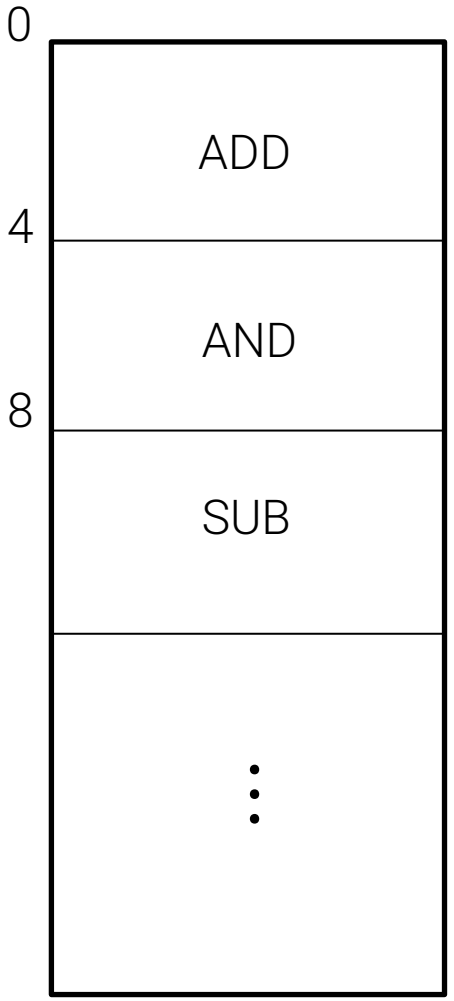
- The use of a microprogram involves placing all control variables in words of ROM for use by the control unit through successive read operations
- The content of the word in ROM at a given address specifies a microinstruction
- Microinstructions generate the microoperations
  - To fetch the instruction from main memory
  - To evaluate the effective address
  - To execute the operation specified by the instruction
  - To return control to the fetch phase in order to repeat the cycle for the next instruction

Microprogrammed Control Unit cnt.

- The microinstruction contains a control word that specifies one or more microoperations
  - Once these operations are executed, the control must determine the next address
- The location of the next microinstruction may be the one next in sequence, or it may be located somewhere else in the control memory
  - For this reason it is necessary to use some bits of the present microinstruction to control the generation of the address of the next microinstruction
- Next address is computed in the next address generator circuit
  - Thus a microinstruction contains bits for initiating microoperations in the data processor part and bits that determine the address sequence for the control memory
  - Next address generator is sometimes called a microprogram sequencer, as it determines the address sequence that is read from control memory

Address Sequencing

- Microinstructions are stored in control memory in groups, with each group specifying a routine
  - Each computer instruction has its own microprogram routine in control memory to generate the microoperations that execute the instruction
  - The hardware that controls the address sequencing of the control memory must be capable of sequencing the microinstructions within a routine and be able to branch from one routine to another
- The steps during the execution of a single computer instruction
  - An initial address is loaded into the control address register when power is turned on in the computer
    - This address is usually the address of the first microinstruction that activates the instruction fetch routine
    - The fetch routine may be sequenced by incrementing the control address register through the rest of its microinstructions. At the end of the fetch routine, the instruction is in the instruction register of the computer
  - The control memory next must go through the routine that determines the effective address of the operand



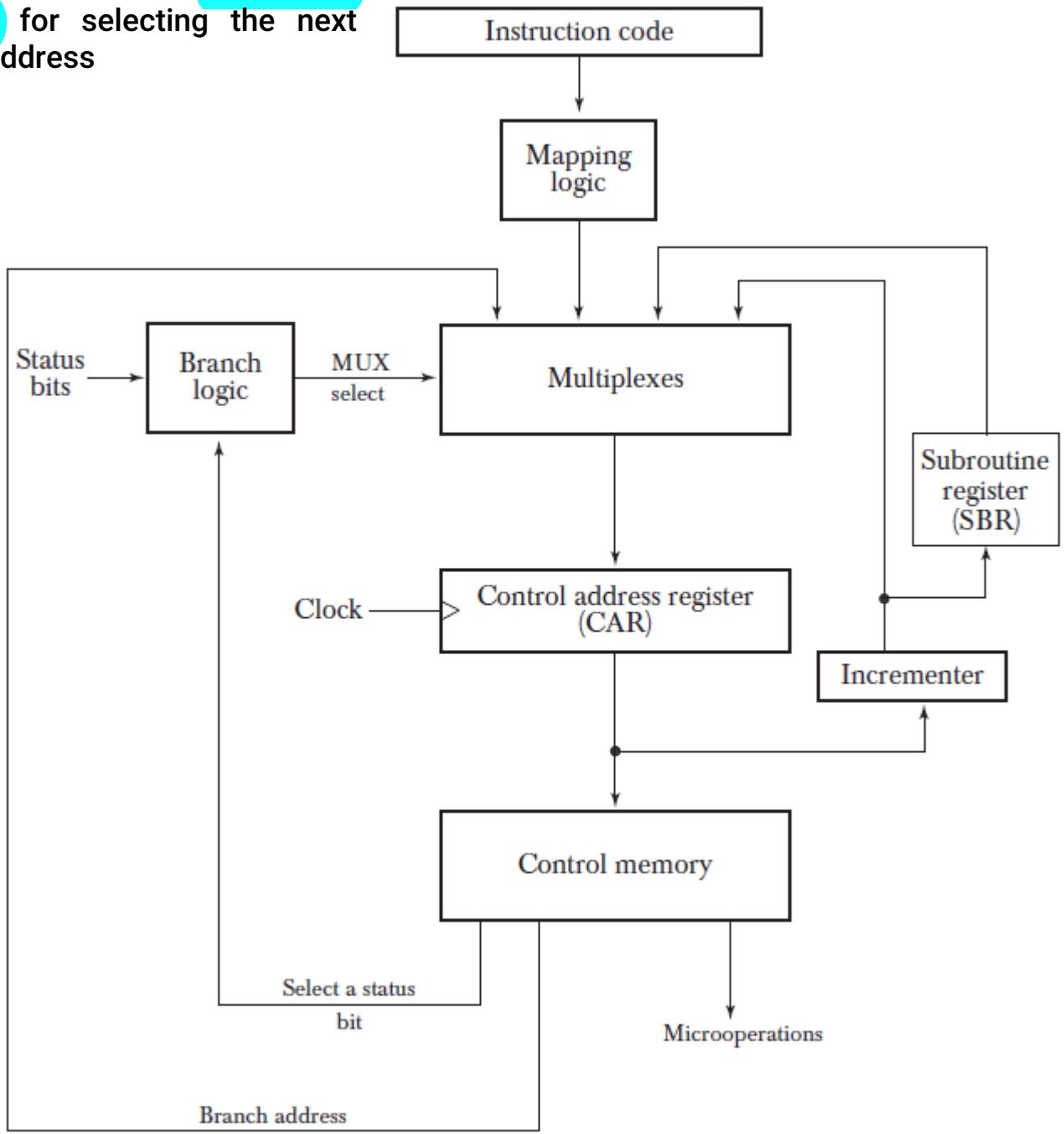
Address Sequencing cnt.

- Effective address computation routine in control memory can be reached through a branch microinstruction
  - Which is conditioned on the status of the mode bits of the instruction
  - When the effective address computation routine is completed, the address of the operand is available in the memory address register
- The next step is to generate the microoperations that execute the instruction fetched from memory
  - The microoperation steps to be generated in processor registers depend on the operation code part of the instruction. Each instruction has its own microprogram routine stored in a given location of control memory
- The transformation from the instruction code bits to an address in control memory where the routine is located is referred to as a mapping

Mapping Procedure

- A mapping procedure is a rule that transforms the instruction code into a control memory address
  - Once the required routine is reached, the microinstructions that execute the instruction may be sequenced by incrementing the control address register
  - But sometimes the sequence of microoperations will depend on values of certain status bits in processor registers
- Microprograms that employ subroutines will require an external register for storing the return address
  - Return addresses cannot be stored in ROM because the unit has no writing capability
- When the execution of the instruction is completed
  - Control must return to the fetch routine
  - This is accomplished by executing an unconditional branch microinstruction to the first address of the fetch routine

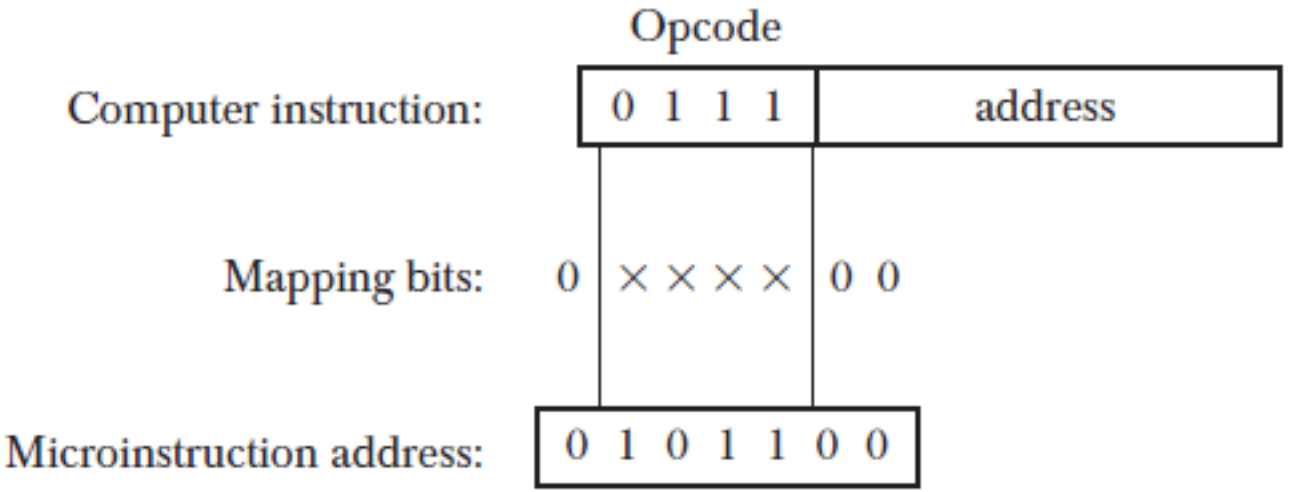
control memory and the associated hardware needed for selecting the next microinstruction address



Mapping from instruction code to microinstruction address

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

The status bits for this type of branch are the bits in the operation code part of the instruction. For example, a computer with a simple instruction format as shown bellow has an operation code of four bits which can specify up to 16 distinct instructions



Mapping of Instruction

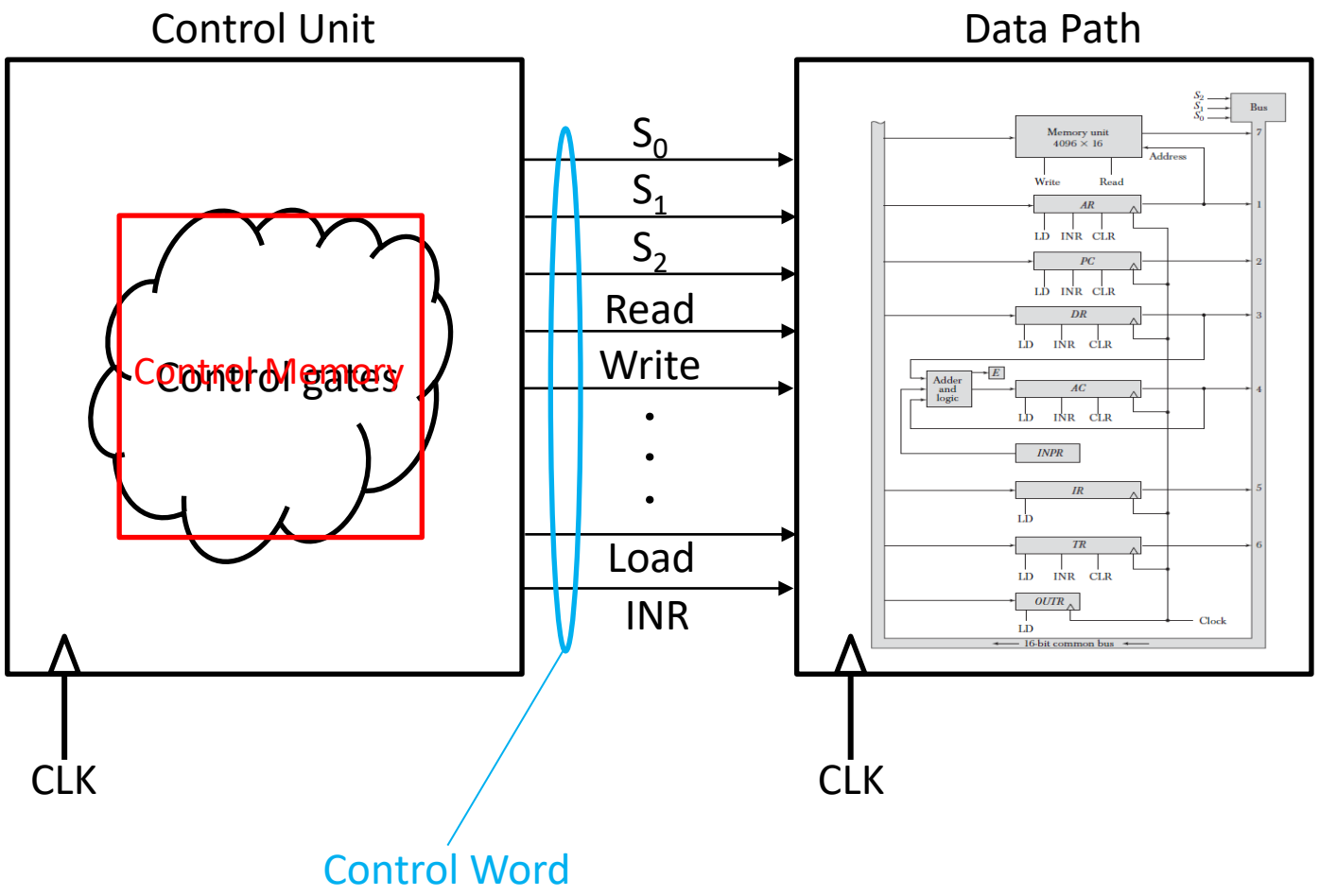
- Assume the control memory has 128 words
  - Requiring an address of seven bits
- For each operation code there exists a microprogram routine in control memory that executes the instruction
- Mapping process
  - Converts the 4-bit operation code to a 7-bit address
    - Placing a 0 in the most significant bit of the address, transferring the four operation code bits, and clearing the two least significant bits
  - Provides for each computer instruction a microprogram routine with a capacity of four microinstructions
    - If the routine needs more than four microinstructions, it can use addresses 1000000 through 1111111
    - If it uses fewer than four microinstructions, the unused memory locations would be available for other routines

Subroutines

- Subroutines are programs that are used by other routines to accomplish a particular task
  - Frequently, many microprograms contain identical sections of code
    - For example, the sequence of microoperations needed to generate the effective address of the operand
      - Common to all memory reference instructions
    - This sequence could be a subroutine
      - That is called from within many other routines to execute the effective address computation
- Provision for storing the return address
  - Accomplished by placing the incremented output from the control address register into a subroutine register
    - Subroutine register can then become the source for transferring the address for the return to the main routine

مرور

Microprogrammed Control Unit

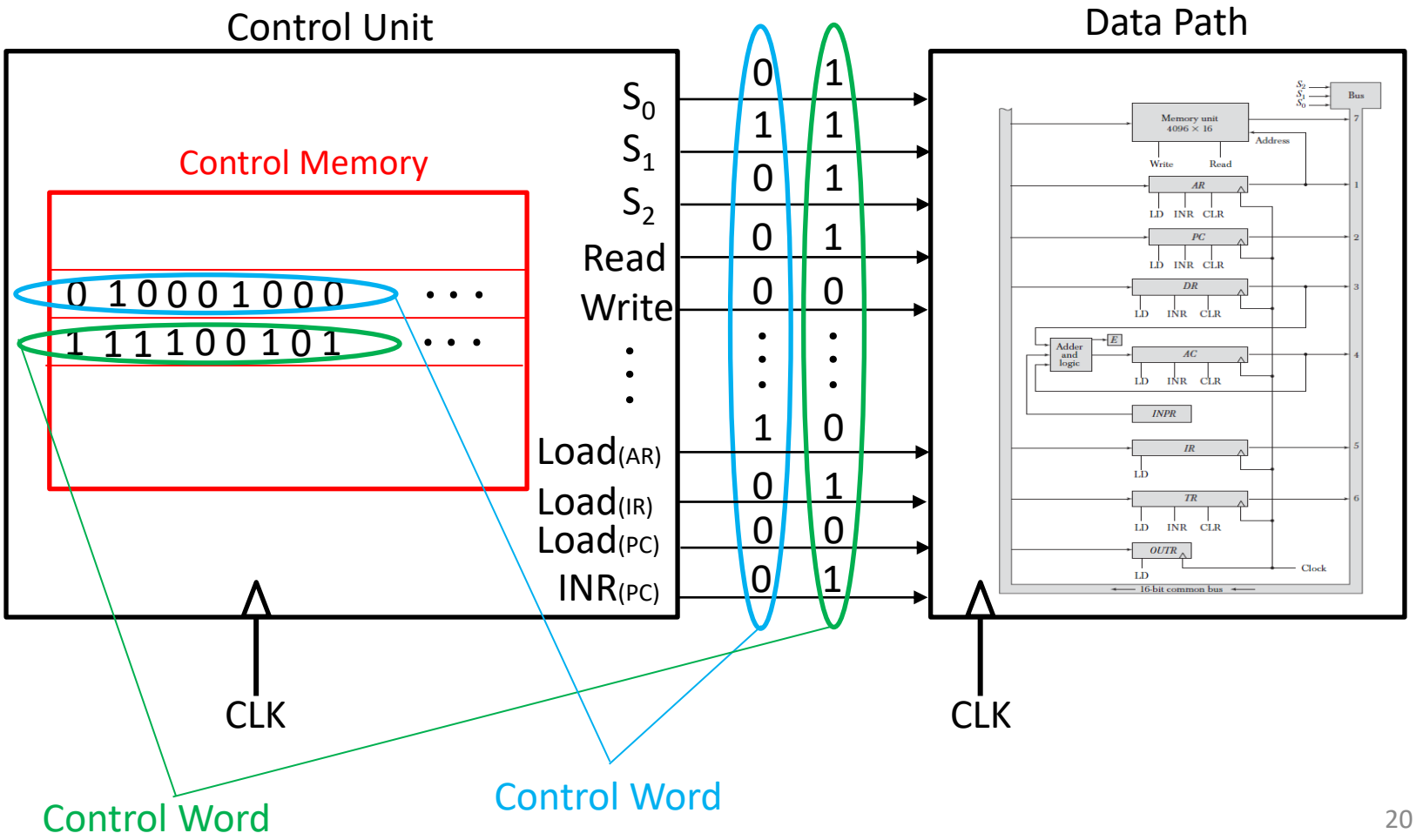


مرور

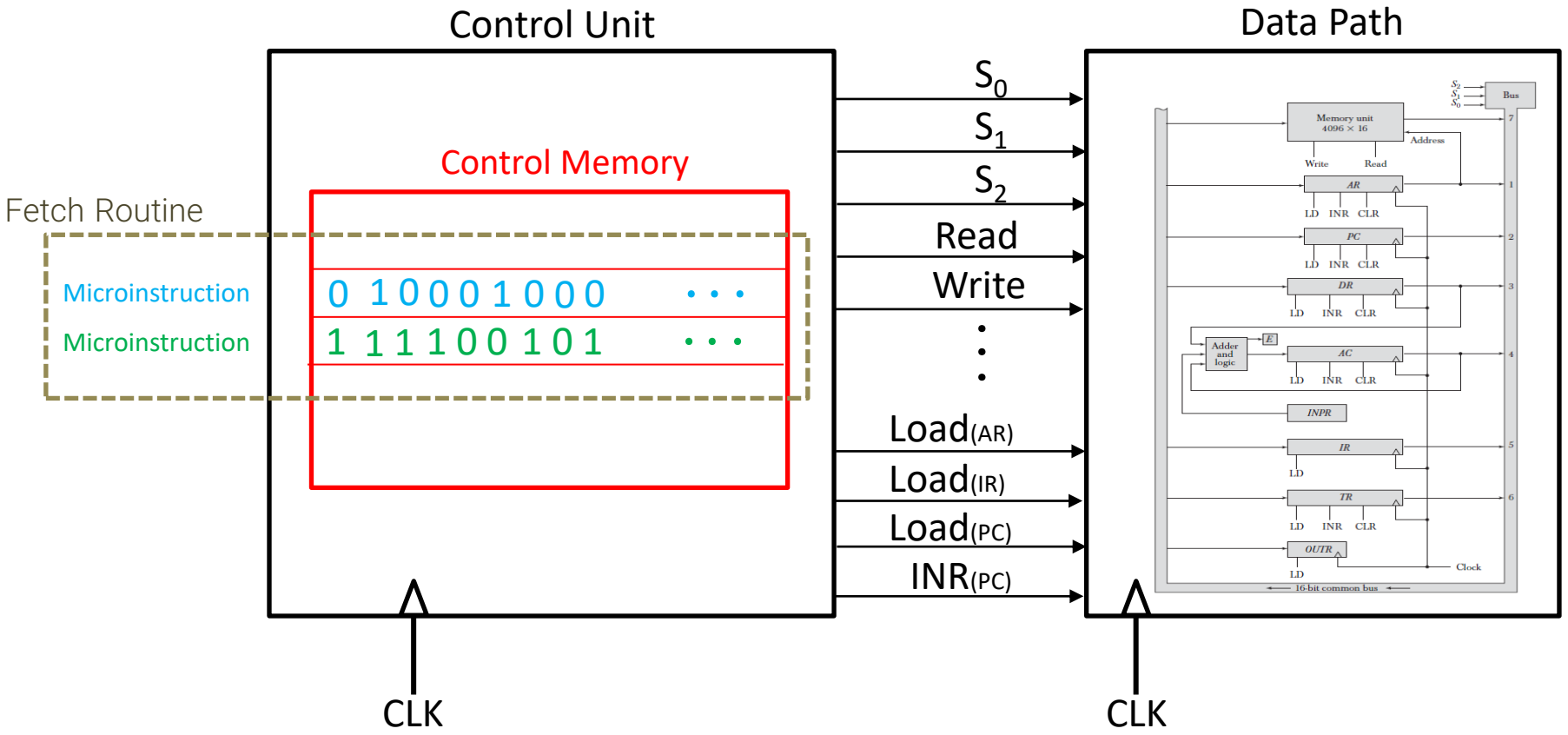
Fetch

T_0: $AR \leftarrow PC$

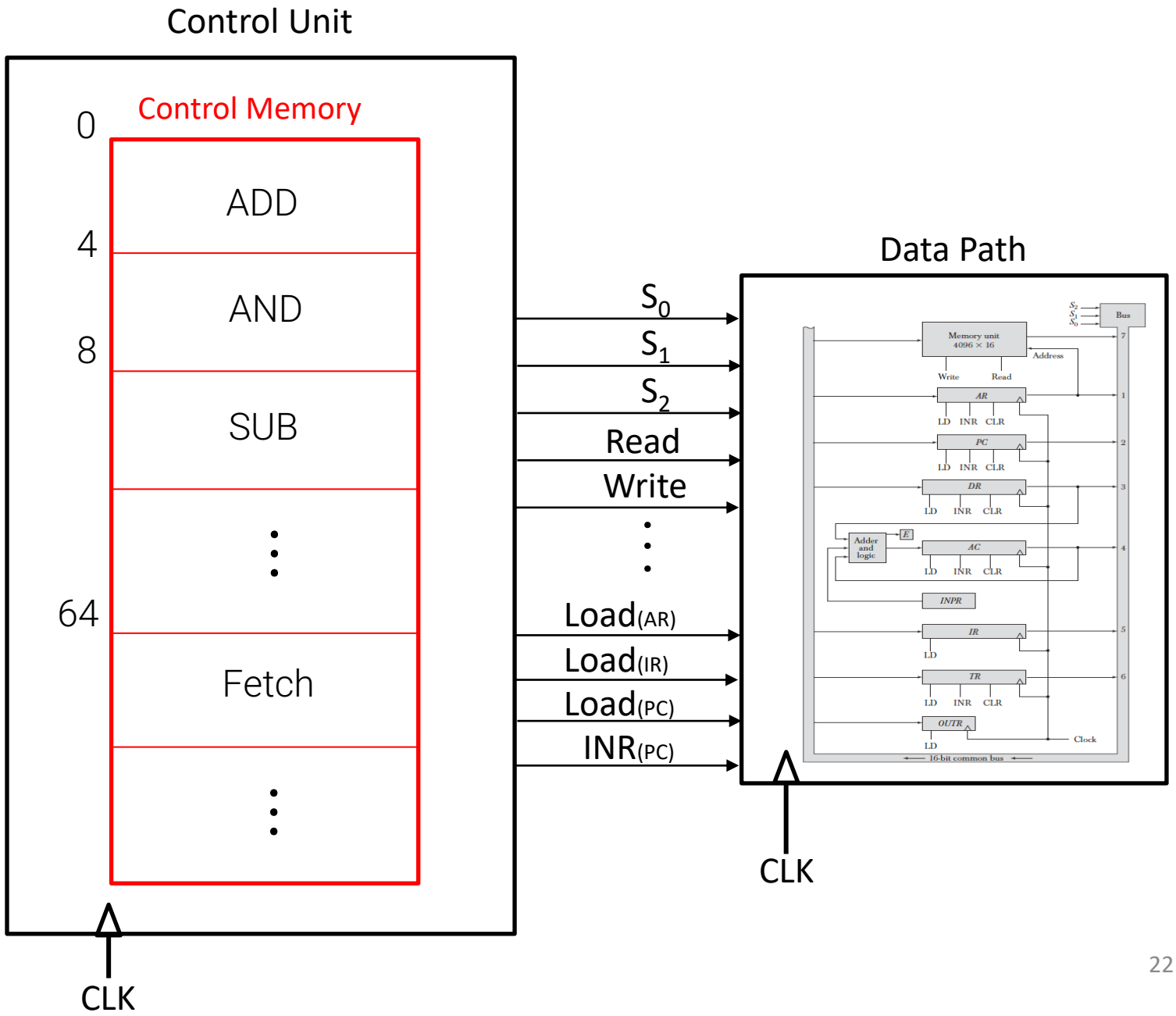
T_1: $IR \leftarrow M[AR],\ PC \leftarrow PC + 1$



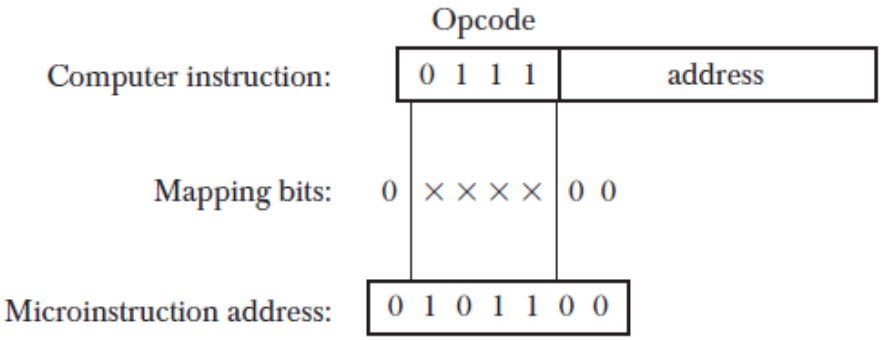
مرور


Control Unit
Data Path
Control Memory
Fetch Routine
Microinstruction
Microinstruction
0 1 0 0 0 1 0 0 0 • • •
1 1 1 1 0 0 1 0 1 • • •
S0
S1
S2
Read
Write
Load(AR)
Load(IR)
Load(PC)
INR(PC)
CLK
CLK
Bus
Memory unit
4096 × 16
Address
Write
Read
AR
PC
DR
Adder and logic
E
AC
INPR
IR
TR
OUTR
LD
INR
CLR
Clock
16-bit common bus

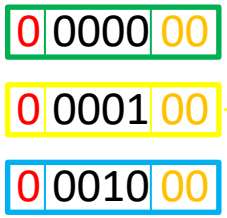
مرور

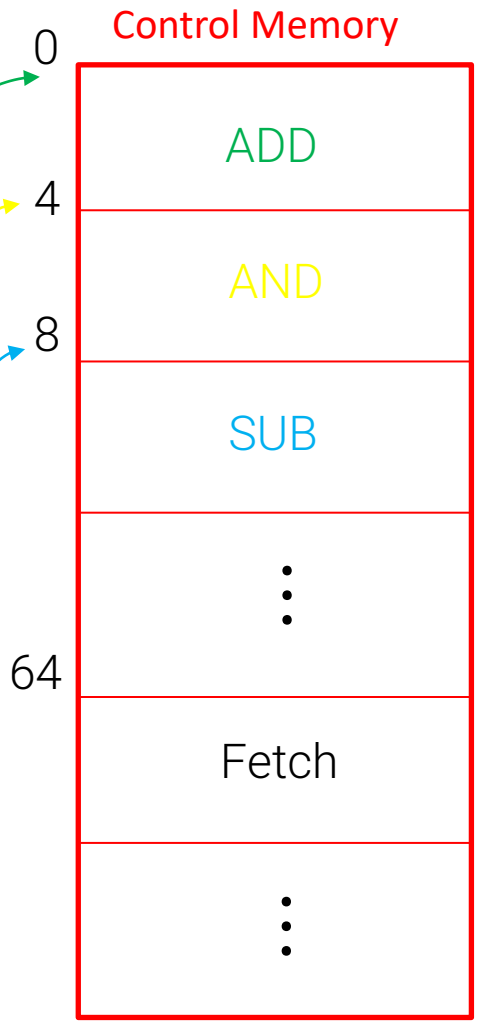

Control Unit
Control Memory
0
ADD
4
AND
8
SUB
64
Fetch
CLK
S_0
S_1
S_2
Read
Write
Load(AR)
Load(IR)
Load(PC)
INR(PC)
Data Path
Bus
Memory unit
4096 × 16
Address
Write
Read
AR
PC
DR
Adder and logic
E
AC
INPR
IR
TR
OUTR
LD
INR
CLR
Clock
16-bit common bus
CLK

مرور

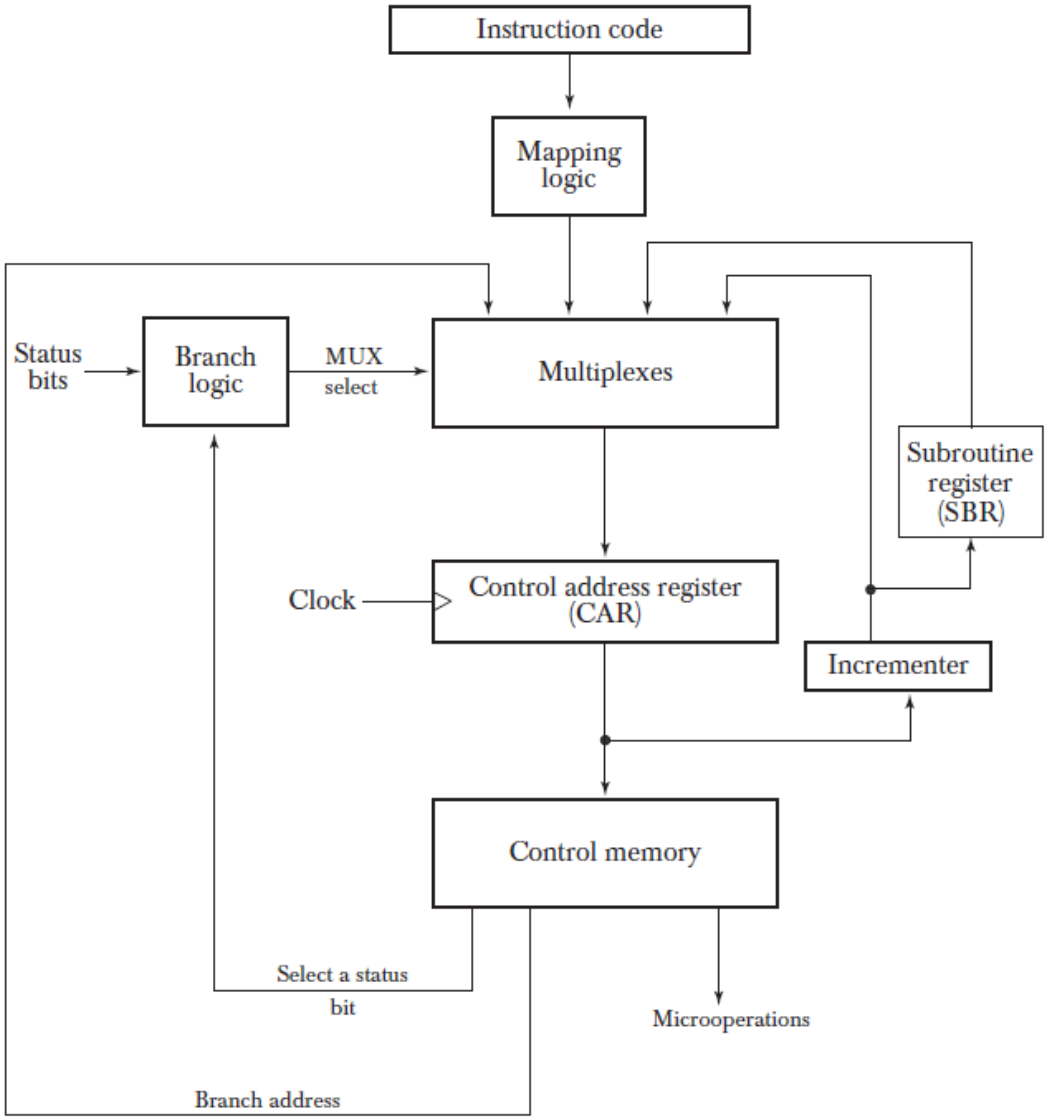


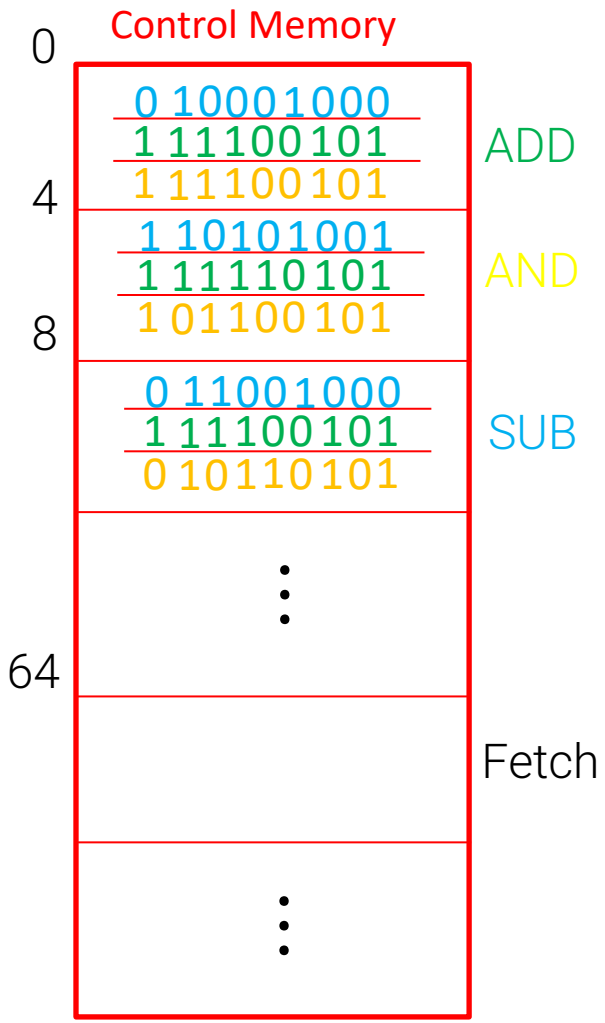
Symbol	Opcode
ADD	0000
AND	0001
SUB	0010
…	…





مرور


Instruction code
Mapping logic
Status bits
Branch logic
MUX select
Multiplexes
Subroutine register (SBR)
Clock
Control address register (CAR)
Incrementer
Control memory
Select a status bit
Microoperations
Branch address

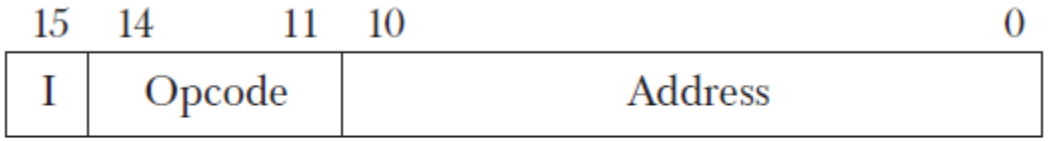

Control Memory
0
4
8
64
0 10001000
1 11100101
1 11100101
ADD
1 10101001
1 11110101
1 01100101
AND
0 11001000
1 11100101
0 10110101
SUB
Fetch

Microprogram Example

Symbol	Opcode	Description
ADD	0000	$AC \leftarrow AC + M[EA]$
BRANCH	0001	If $(AC < 0)$ then $(PC \leftarrow EA)$
STORE	0010	$M[EA] \leftarrow AC$
EXCHANGE	0011	$AC \leftarrow M[EA], M[EA] \leftarrow AC$

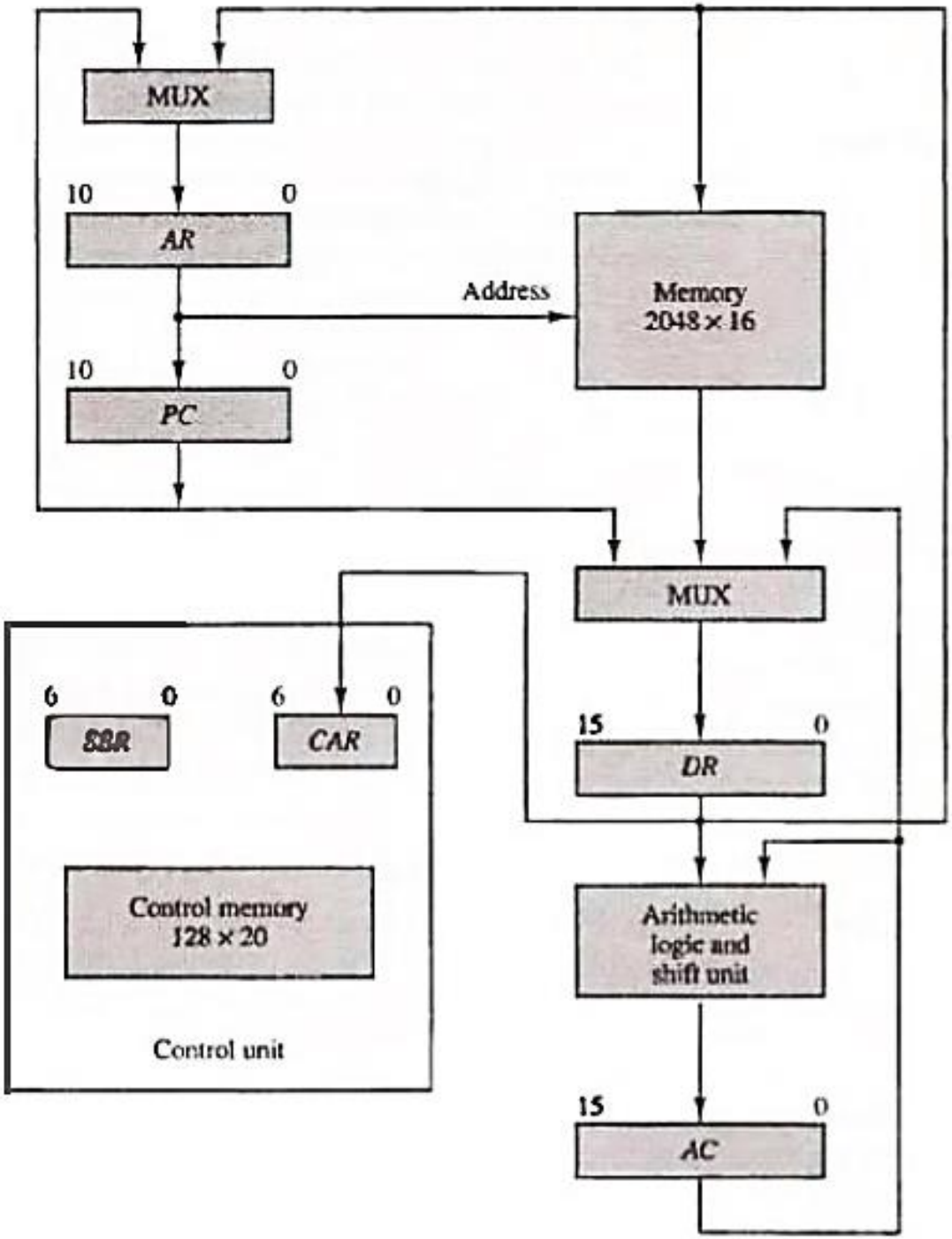
EA is the effective address

- Computer instruction format
  - Consists of three fields
    - A 1-bit field for indirect addressing symbolized by I
    - A 4-bit operation code (opcode)
    - An 11-bit address field



Instruction format

Computer hardware



Data Path

- The transfer of information among the registers in the processor is done through multiplexers rather than a common bus
  - DR can receive information from AC, PC, or memory
  - AR can receive information from PC or DR
  - PC can receive information only from AR

Microinstruction Format

- 20 bits of the microinstruction are divided into four functional parts
  - The three fields Fl, F2, and F3 specify microoperations for the computer
  - The CD field selects status bit conditions
  - The BR field specifies the type or branch to be used
  - The AD field contains a branch address
    - The address field is seven bits wide, since the control memory has $128 = 2^7$ words
- Microoperations are subdivided into three fields of three bits each
  - The three bits in each field are encoded to specify seven distinct microoperations
  - A total of 21 microoperations
  - No more than three microoperations can be chosen for a microinstruction, one from each field
  - If fewer than three microoperations are used, one or more of the fields will use the binary code 000 for no operation

Code for Microinstruction Fields

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

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

F3	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow$ shl AC	SHL
100	$AC \leftarrow$ shr AC	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

Code for Microinstruction Fields

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

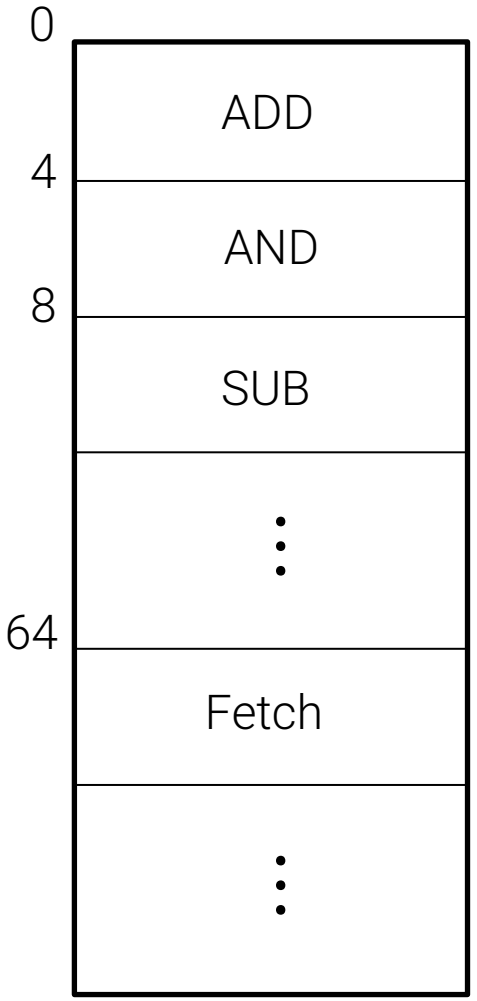
BR	Symbol	Function
00	JMP	$CAR \leftarrow AD$ if condition = 1
		$CAR \leftarrow CAR + 1$ if condition = 0
01	CALL	$CAR \leftarrow AD$, $SBR \leftarrow CAR + 1$ if condition = 1
		$CAR \leftarrow CAR + 1$ if condition = 0
10	RET	$CAR \leftarrow SBR$ (Return from subroutine)
11	MAP	$CAR(2-5) \leftarrow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

Symbolic Microinstructions

- Each symbolic microinstruction is divided into five fields
  - Label, microoperations, CD, BR, and AD
    - The label field may be empty or it may specify a symbolic address. A label is terminated with a colon (:)
    - The microoperations field consists of one, two, or three symbols, separated by commas
    - The CD field has one of the letters U, I, S, or Z
    - The BR field contains one of the four symbols defined in the Table
    - The AD field specifies a value for the address field of the microinstruction in one of three possible ways:
      - With a symbolic address, which must also appear as a label
      - With the symbol NEXT to designate the next address in sequence
      - When the BR field contains a RET or MAP symbol, the AD field is left empty and is converted to seven zeros by the assembler
    - Pseudoinstruction ORG
      - To define the origin, or first address, of a microprogram routine

Fetch Routine

- The control memory has 128 words, and each word contains 20 bits
  - To microprogram the control memory, it is necessary to determine the bit values of each of the 128 words
  - First 64 words (addresses 0 to 63)
    - Occupied by the routines for the 16 instructions
  - The last 64 words may be used for any other purpose
    - A convenient starting location for the fetch routine is address 64



Fetch Routine

$$AR \leftarrow PC$$
$$DR \leftarrow \mathrm{M}[AR], \quad PC \leftarrow PC + 1$$
$$AR \leftarrow DR(0\text{–}10), \quad CAR(2\text{–}5) \leftarrow DR(11\text{–}14), \; CAR(0,1,6) \leftarrow 0$$

- The fetch routine needs three microinstructions
  - Placed in control memory at addresses 64, 65, and 66

```
            ORG  64
FETCH:      PCTAR              U     JMP     NEXT
            READ, INCPC        U     JMP     NEXT
            DRTAR              U     MAP
```

Binary Address	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

Symbolic Microprogram

- Execution of MAP microinstruction in the fetch routine
  - A branch to address 0xxxx00, where xxxx are the four bits of the operation code
  - For example, ADD instruction
    - Operation code is 0000
      - The MAP microinstruction will transfer to CAR the address 0000000
      - Which is the start address for the ADD routine in control memory
  - The first address for the BRANCH and STORE routines
    - 0 0001 00 (decimal 4) and 0 0010 00 (decimal 8)
  - The first address for the other 13 routines are at address values 12, 16, 20, . . . , 60
    - This gives four words in control memory for each routine.

Symbolic Microprogram

- In each routine we must provide microinstructions for evaluating the effective address and for executing the instruction
  - The indirect address mode is associated with all memory-reference instructions
  - A saving in the number of control memory words may be achieved if the microinstructions for the indirect address are stored as a subroutine
    - This subroutine, symbolized by INDRCT, is located right after the fetch routine
  - Control memory words at locations 69 to 127 have not been used
  - Instructions such as multiply, divide, and others that require a long sequence of microoperations
    - Will need more than four microinstructions for their execution.
    - Control memory words 69 to 127 can be used for this purpose

Symbolic Microprogram

Label	Microoperations	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

Symbolic Microprogram

```
        ORG 64
FETCH:  PCTAR          U    JMP    NEXT
        READ, INCPC    U    JMP    NEXT
        DRTAR          U    MAP
INDRCT: READ           U    JMP    NEXT
        DRTAR          U    RET
```

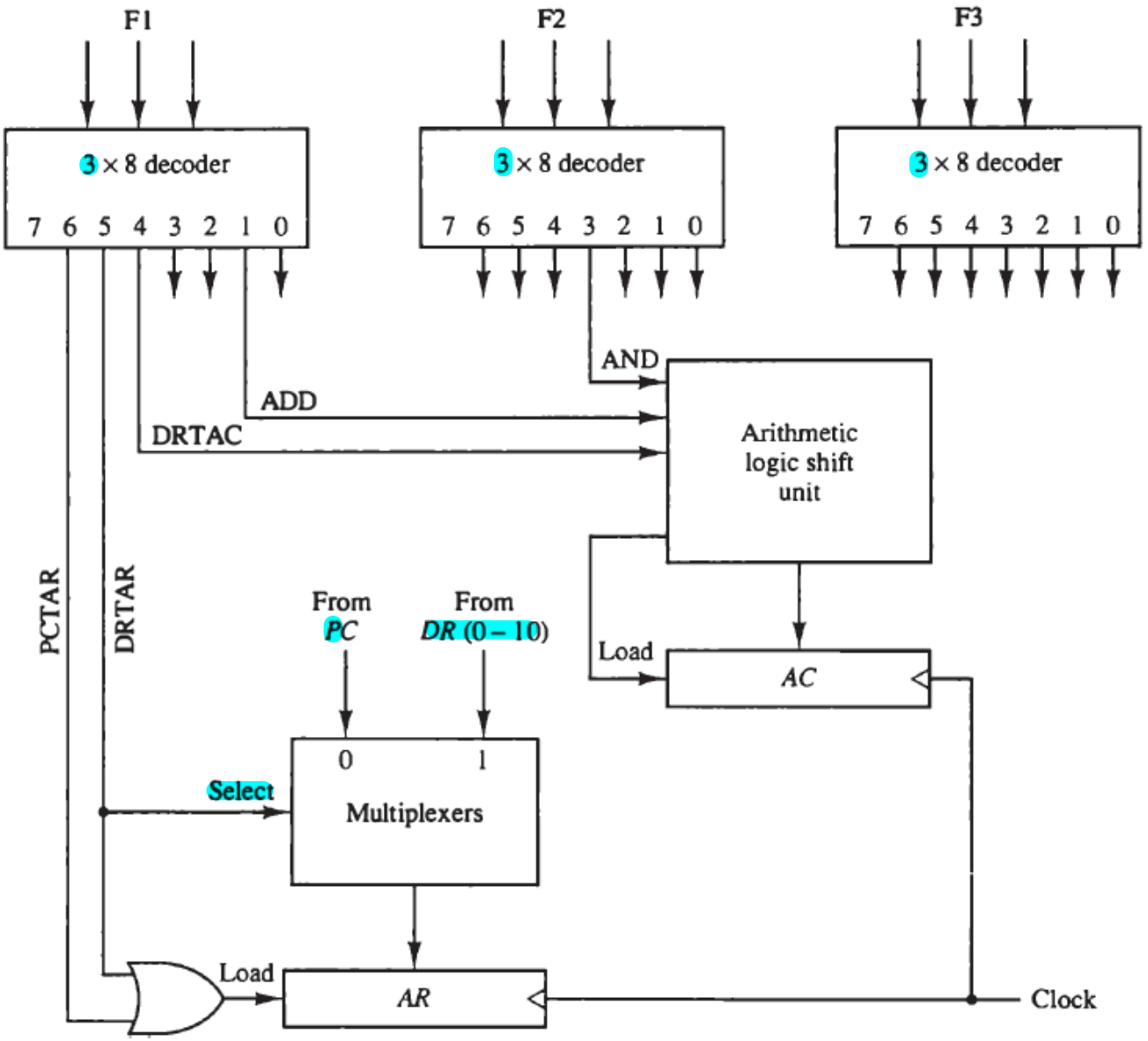
Binary Microprogram

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

Design of Control Unit

- The bits of the microinstruction are divided into fields
  - each field defining a distinct, separate function
- The number of control bits that initiate microoperations can be reduced
  - By grouping mutually exclusive variables into fields and encoding the k bits in each field to provide 2^k microoperations
  - Each field requires a decoder to produce the corresponding control signals
  - This method reduces the size of the microinstruction bits
    - But requires additional hardware external to the control memory
    - Increases the delay time of the control signals
      - Because they must propagate through the decoding circuits

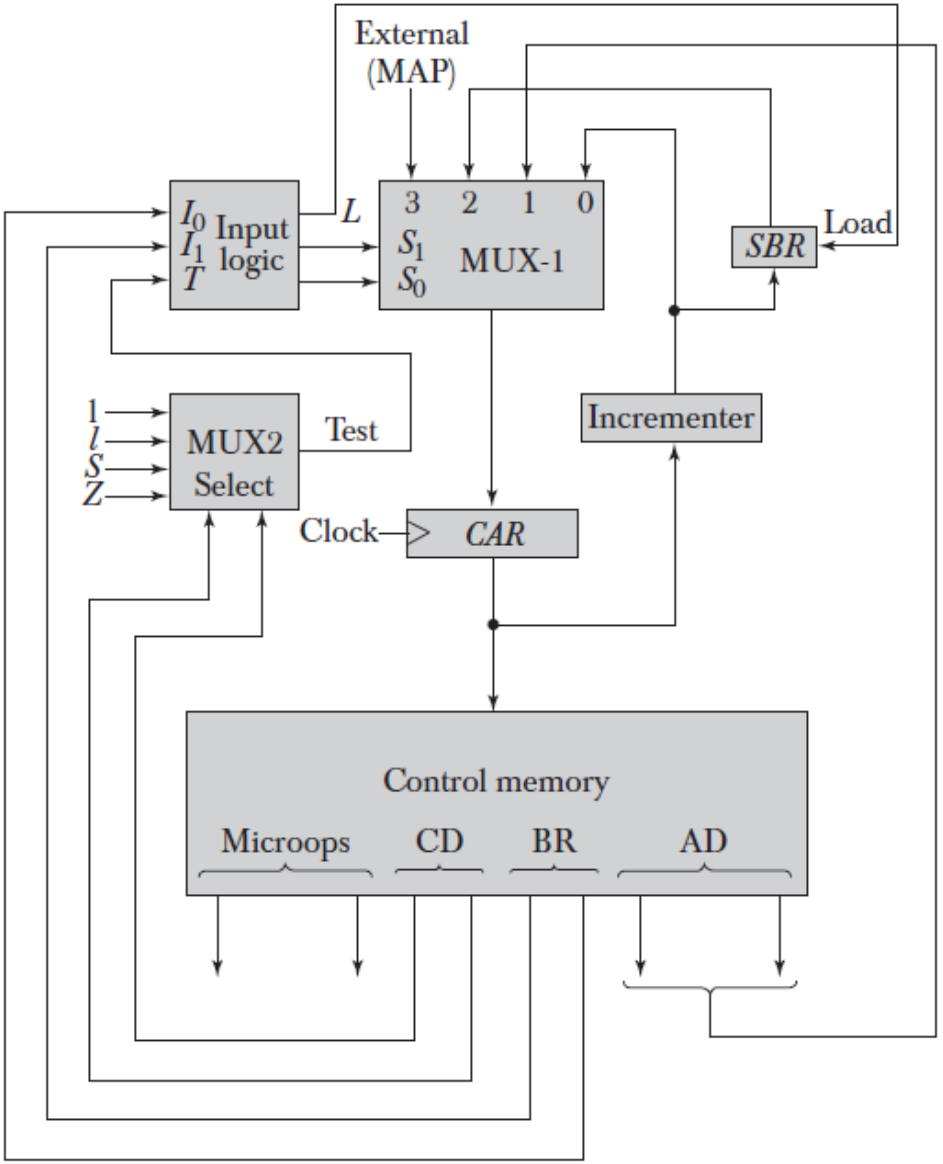
Design of Control Unit cnt.



Microprogram Sequencer

- Basic components of a microprogrammed control unit
  - The control memory and the circuits that select the next address
- The address selection part is called a microprogram sequencer
  - The next-address logic of the sequencer determines the specific address source to be loaded into the control address register
  - The choice of the address source is guided by the next-address information bits that the sequencer receives from the present microinstruction.
- Microprogram sequencer
  - There are two multiplexers in the circuit
    - The first multiplexer selects an address from one of four sources and routes it into a control address register CAR
    - The second multiplexer tests the value of a selected status bit and the result of the test is applied to an input logic circuit

Microprogram Sequencer Diagram



BR Field		Input I_1	I_0	T	MUX 1 S_1	S_0	Load *SBR* L
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	1	0	1	0	0	0	0
0	1	0	1	1	1	1	1
1	0	1	0	×	1	0	0
1	1	1	1	×	1	1	0

پایان

موفق و پیروز باشید