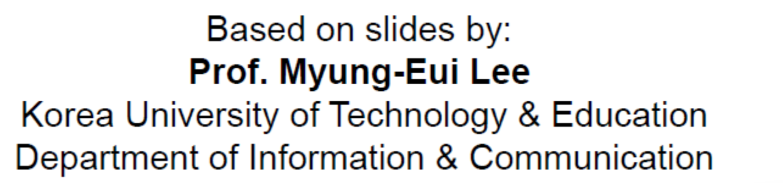
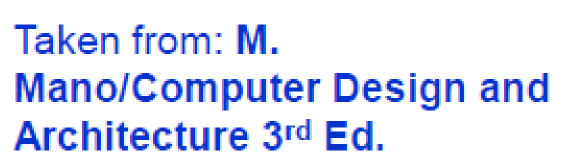
Computer Architecture Lec 5a

Dr. Esti Stein

(Partly taken from Dr. Alon Schclar slides)

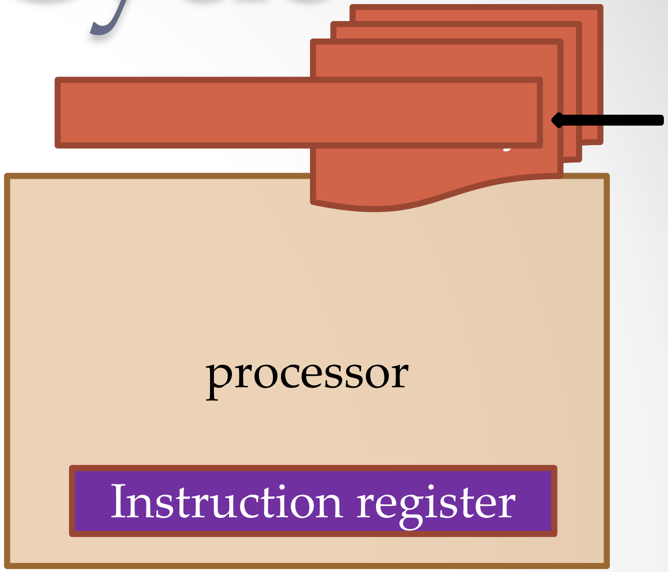

Based on slides by:
Prof. Myung-Eui Lee
Korea University of Technology & Education
Department of Information & Communication


Taken from: **M. Mano/Computer Design and Architecture 3rd Ed.**

Instruction Cycle

Instruction Fetch

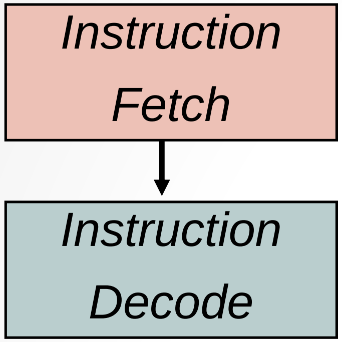
Obtain instruction from program storage in memory



processor

Instruction register

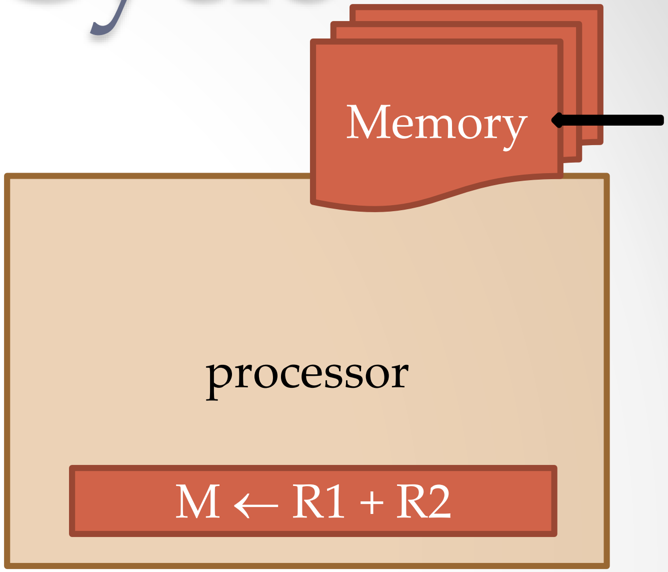
Instruction Cycle


Instruction Fetch

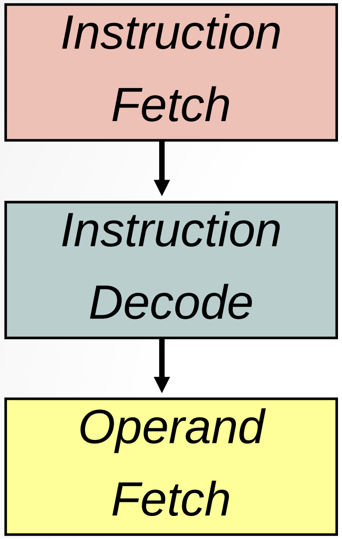
Instruction Decode

Obtain instruction from program storage in memory

Determine required actions and instruction size



Instruction Cycle



Instruction Fetch

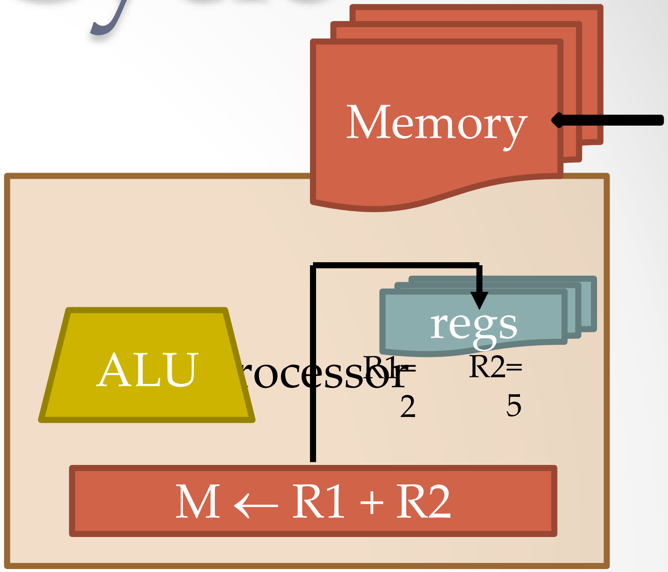
Obtain instruction from program storage in memory

Instruction Decode

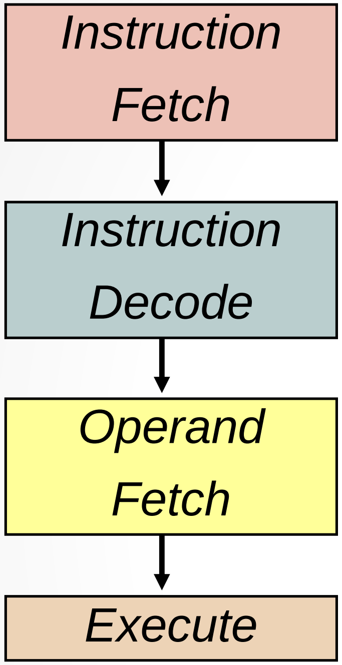
Determine required actions and instruction size

Operand Fetch

Locate and obtain operand data



Instruction Cycle



Instruction Fetch

Obtain instruction from program storage in memory

Instruction Decode

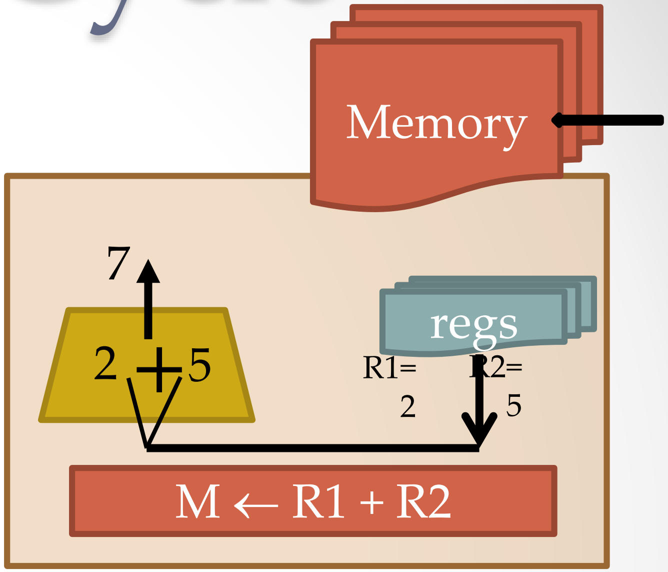
Determine required actions and instruction size

Operand Fetch

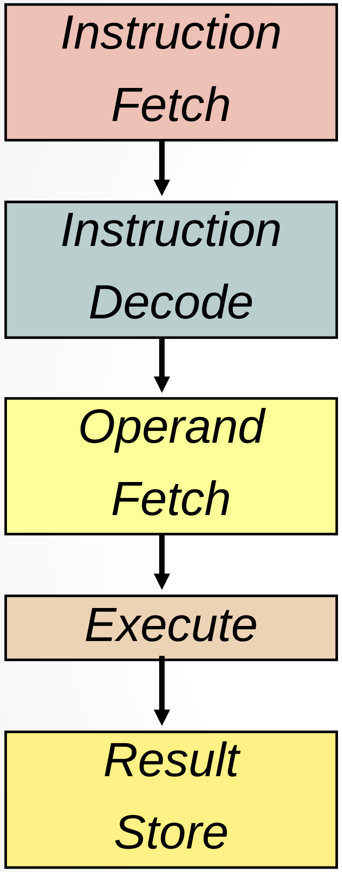
Locate and obtain operand data

Execute

Compute result value or status



Instruction Cycle



Instruction Fetch

Obtain instruction from program storage in memory

Instruction Decode

Determine required actions and instruction size

Operand Fetch

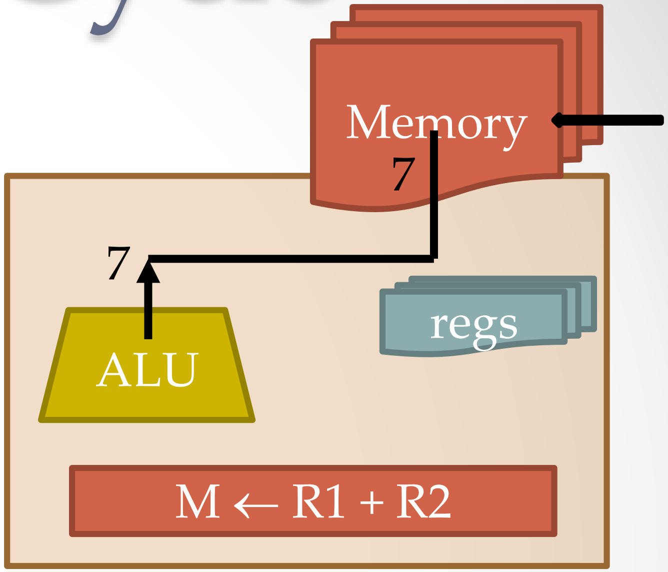
Locate and obtain operand data

Execute

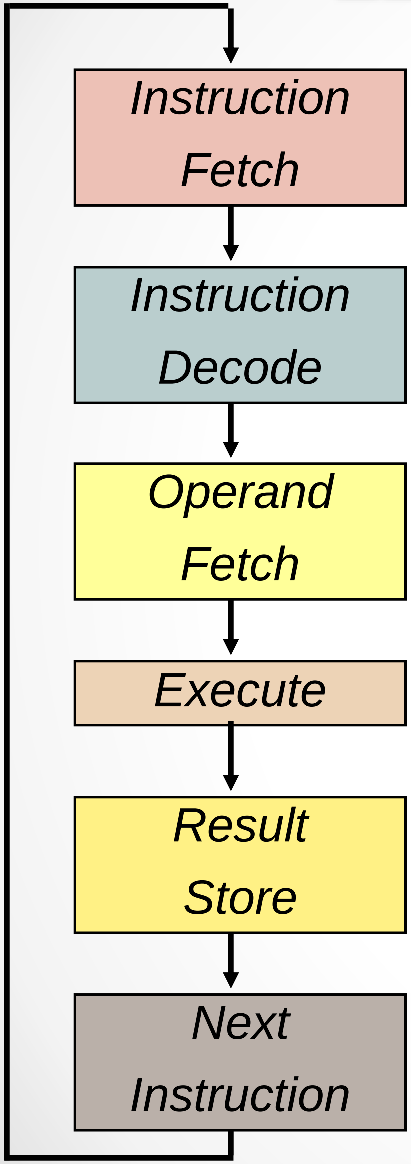
Compute result value or status

Result Store

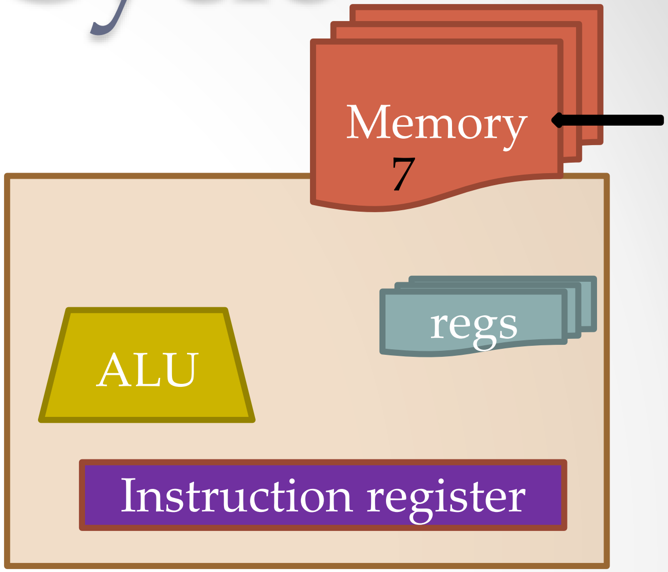
Deposit results in storage



Instruction Cycle



Instruction Fetch	Obtain instruction from program storage in memory
Instruction Decode	Determine required actions and instruction size
Operand Fetch	Locate and obtain operand data
Execute	Compute result value or status
Result Store	Deposit results in storage
Next Instruction	Determine next instruction (not the next in case of **branch**)

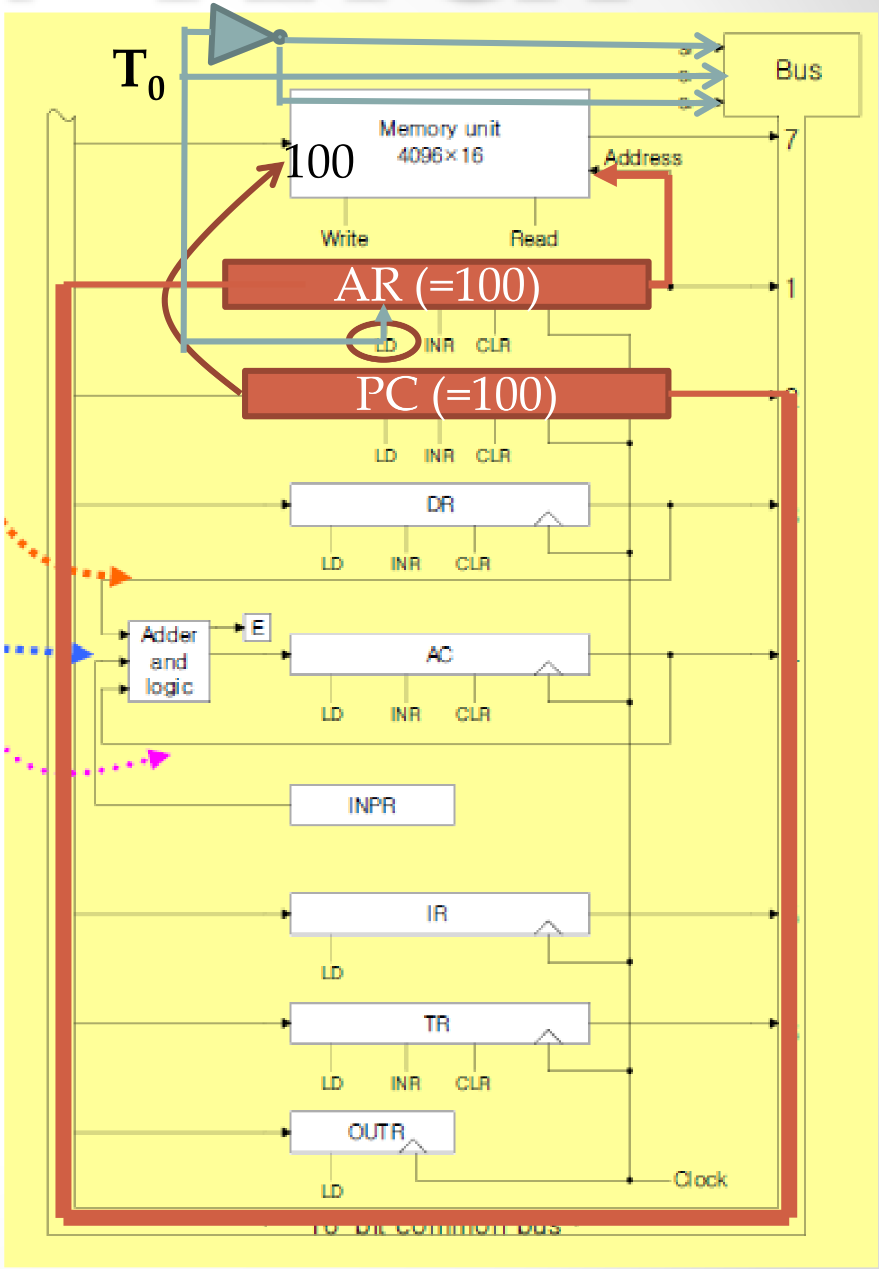


Instruction - FETCH

Instruction Fetch

Obtain instruction from program storage in memory

$$\boxed{T_0 : AR \leftarrow PC}$$



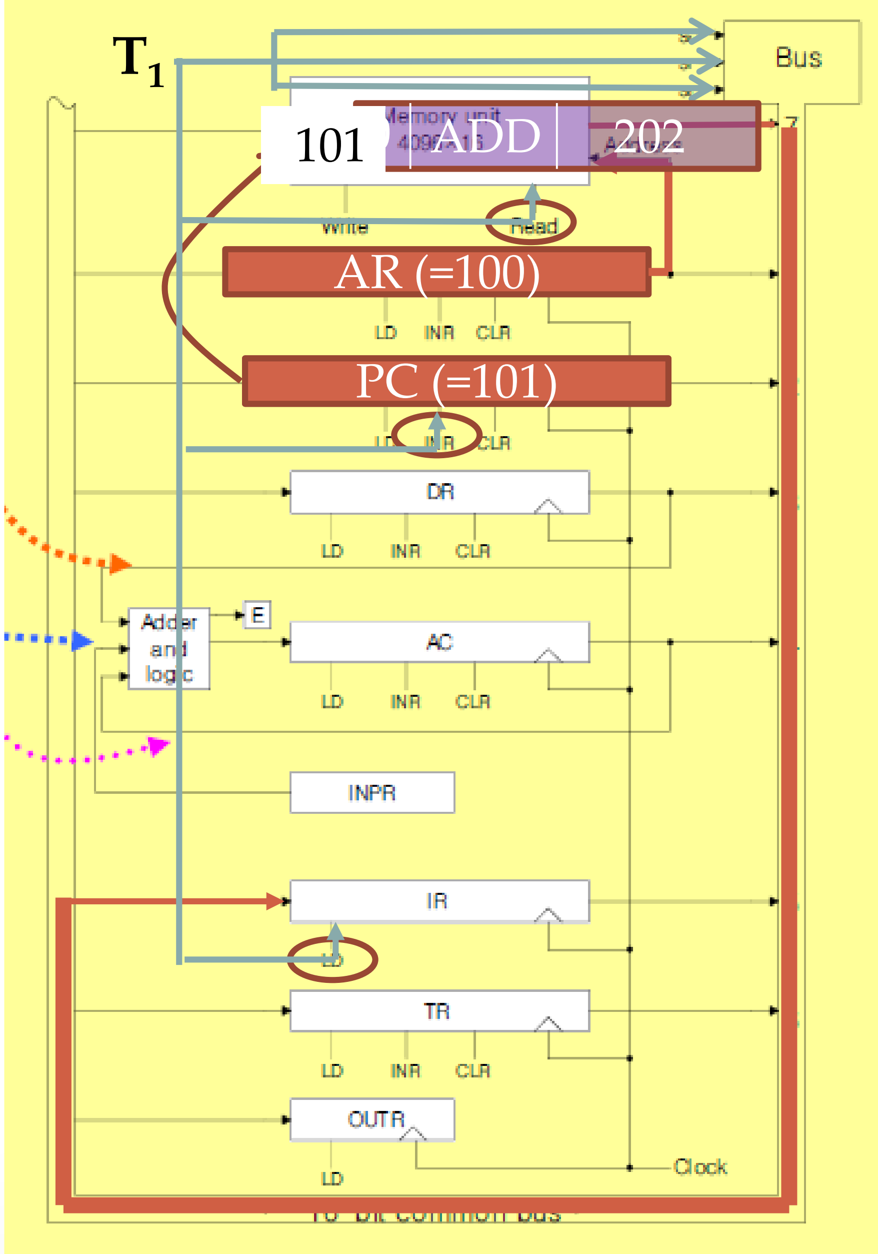
Instruction - FETCH

Instruction Fetch

Obtain instruction from program storage in memory

$$T_0 : AR \leftarrow PC$$

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



Instruction - DECODE

Instruction Fetch

↓

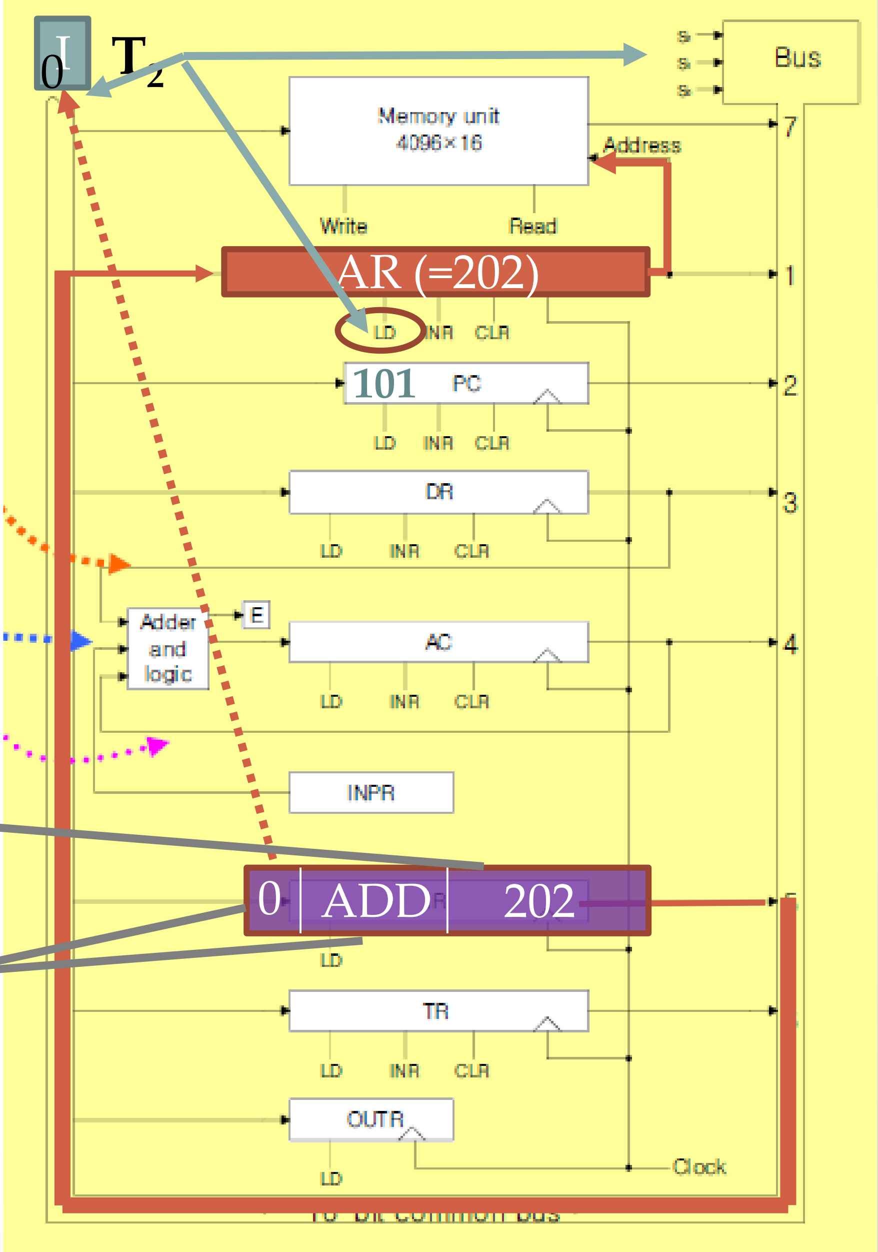
Instruction Decode

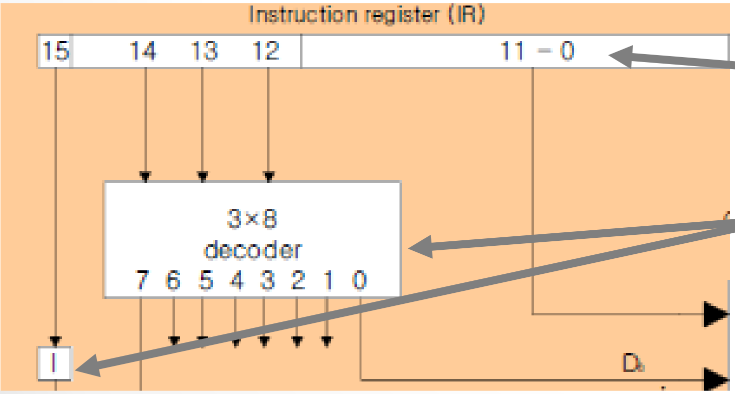
$T_0 : AR \leftarrow PC$

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

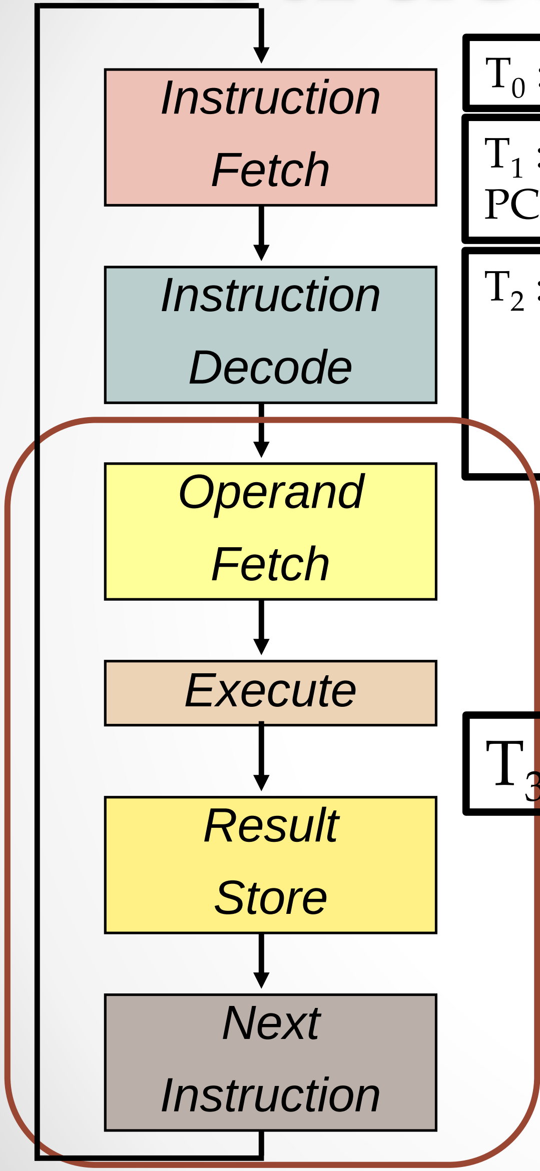
Determine required actions and instruction size

$$\begin{aligned} T_2 : D_0 ,.., D_7 &\leftarrow \text{Decode } IR(12\text{-}14) \\ AR &\leftarrow IR(0\text{-}11), \\ I &\leftarrow IR(15) \end{aligned}$$





Instruction - EXECUTION

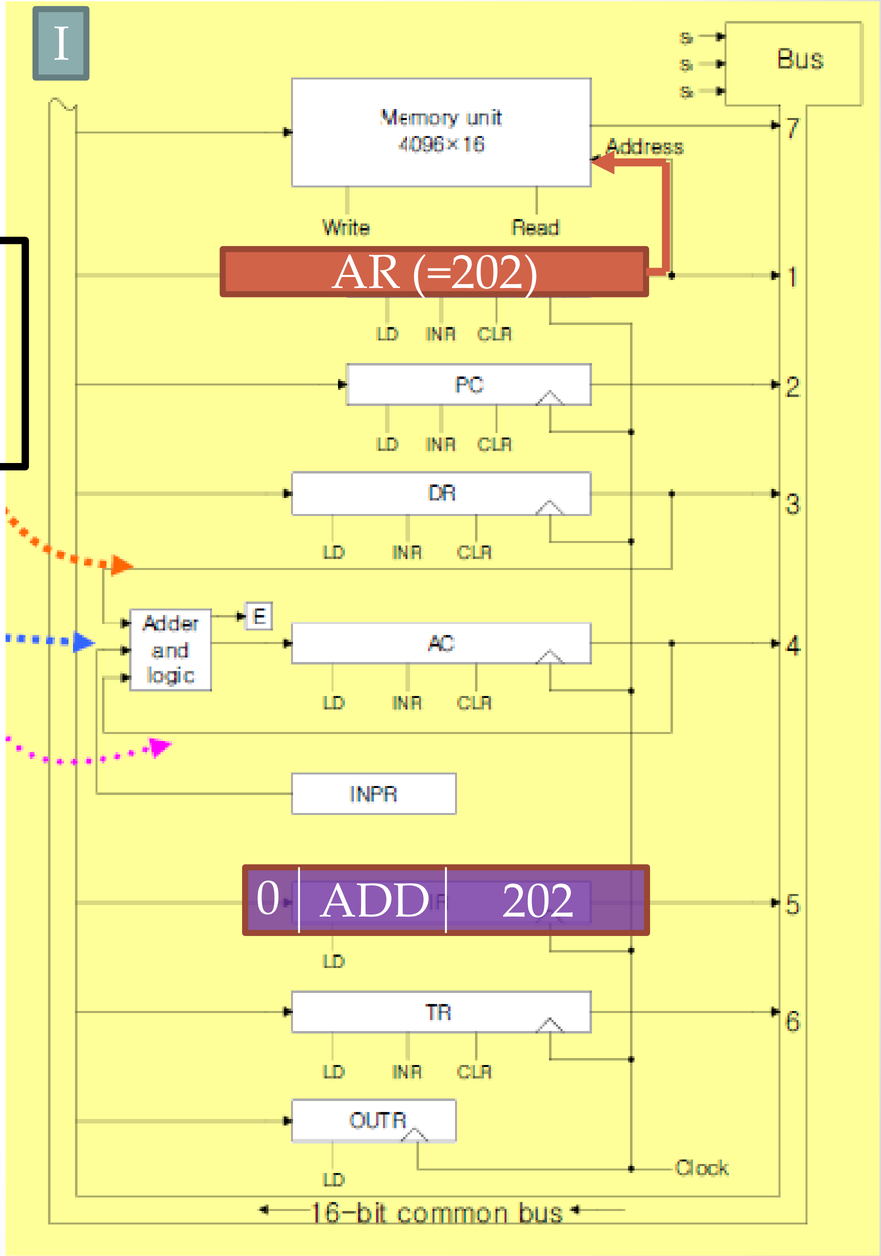


$T_0 : AR \leftarrow PC$

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

$T_2 : D_0 ,..., D_7 \leftarrow$ Decode IR(12-14)
$AR \leftarrow IR(0\text{-}11),$
$I \leftarrow IR(15)$

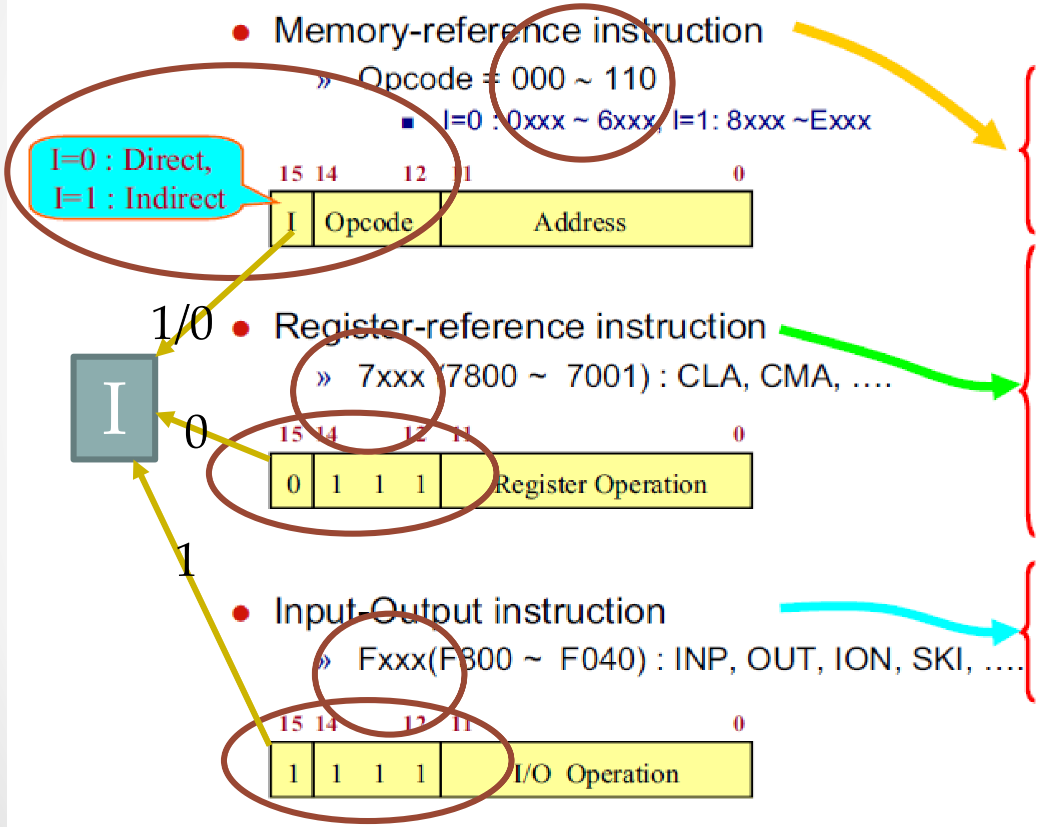
$T_3\ T_4\ T_5\ T_6$



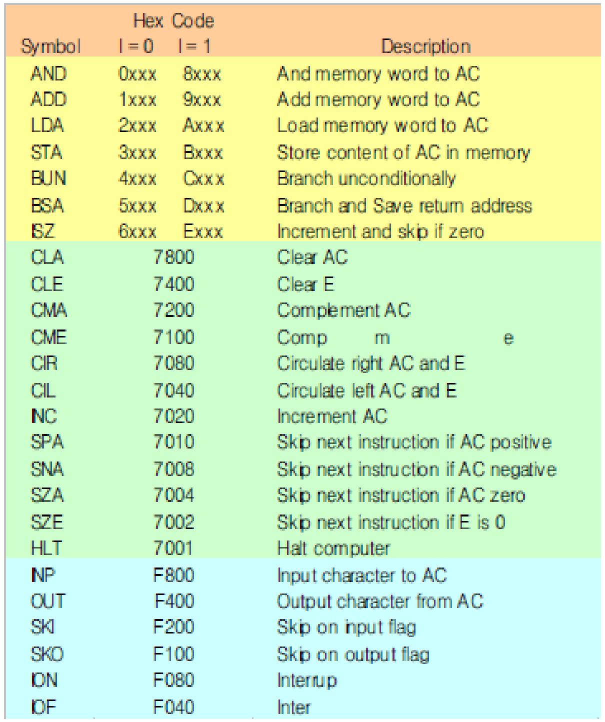
The Instruction Format

5-3 Computer Instruction

3 Instruction Code Formats : *Fig. 5-5*

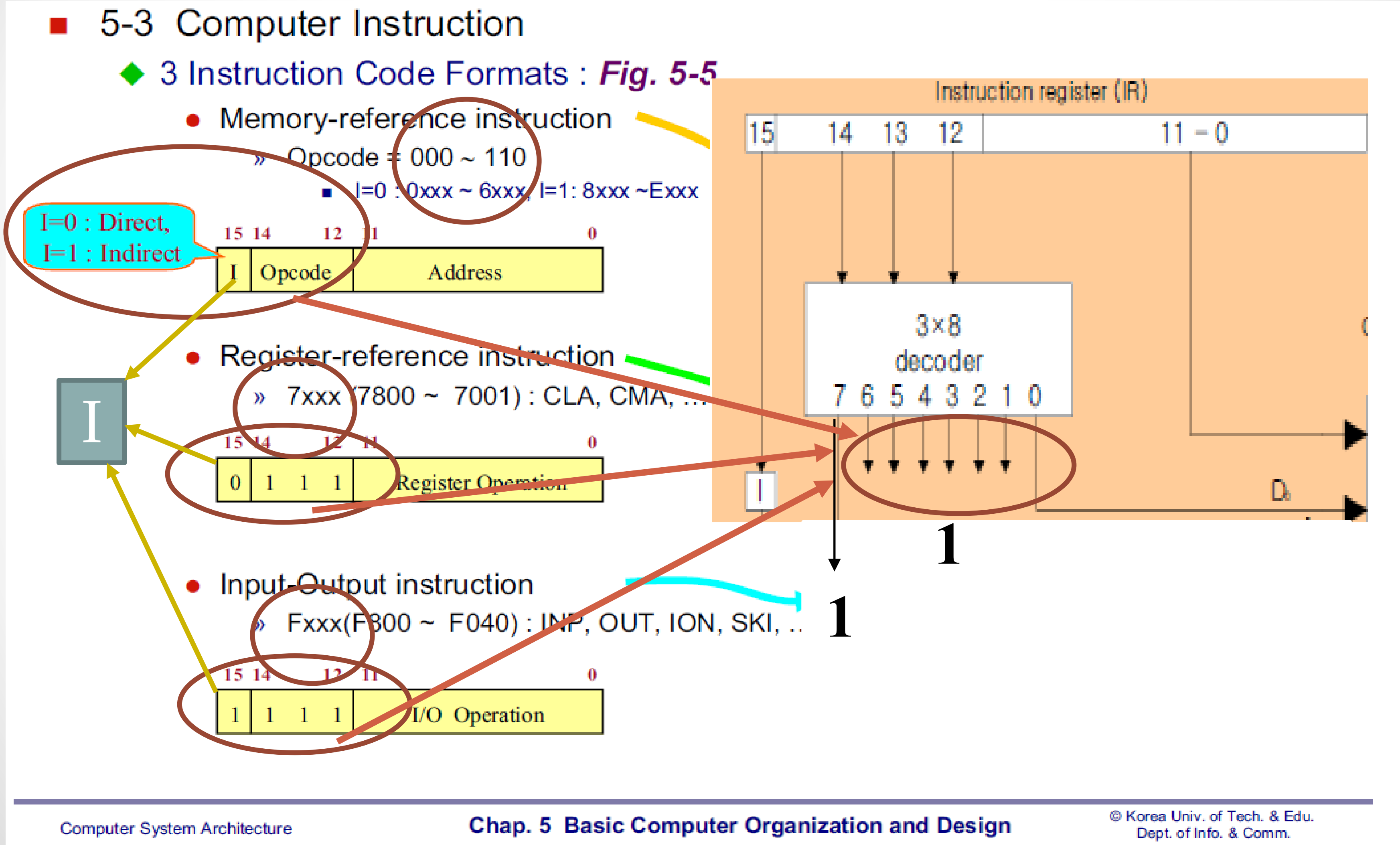


- Memory-reference instruction
  - » Opcode = 000 ~ 110
    - I=0 : 0xxx ~ 6xxx, I=1: 8xxx ~Exxx
- Register-reference instruction
  - » 7xxx (7800 ~ 7001) : CLA, CMA, ….
- Input-Output instruction
  - » Fxxx(F800 ~ F040) : INP, OUT, ION, SKI, ….



Symbol	Hex Code I = 0	 I = 1	Description
AND	0xxx	8xxx	And memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load memory word to AC
STA	3xxx	Bxxx	Store content of AC in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and Save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Comp m e
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instruction if AC positive
SNA	7008		Skip next instruction if AC negative
SZA	7004		Skip next instruction if AC zero
SZE	7002		Skip next instruction if E is 0
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrup
IOF	F040		Inter

The Instruction Format



5-3 Computer Instruction

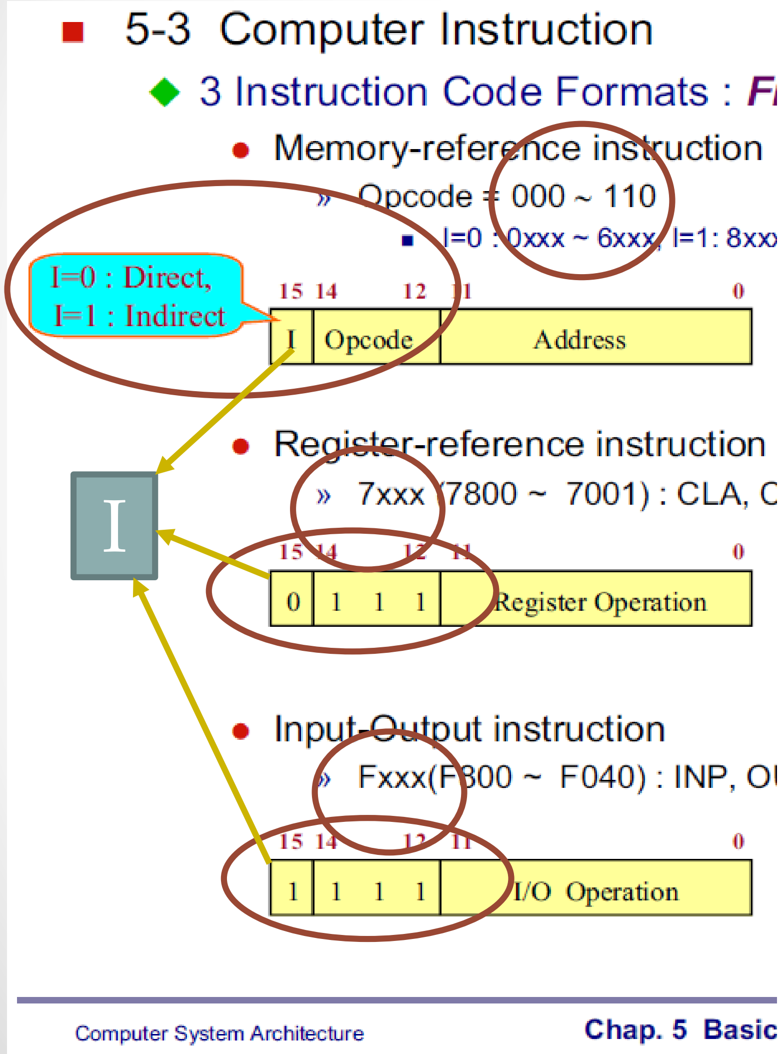
- 3 Instruction Code Formats : ***Fig. 5-5***
  - Memory-reference instruction
    - » Opcode = 000 ~ 110
      - I=0 : 0xxx ~ 6xxx, I=1: 8xxx ~Exxx
  - Register-reference instruction
    - » 7xxx (7800 ~ 7001) : CLA, CMA, ...
  - Input-Output instruction
    - » Fxxx(F800 ~ F040) : INP, OUT, ION, SKI, ..

Computer System Architecture

Chap. 5 Basic Computer Organization and Design

© Korea Univ. of Tech. & Edu.
Dept. of Info. & Comm.

Determine Instruction Type



■ 5-3 Computer Instruction

◆ 3 Instruction Code Formats : Fi

- Memory-reference instruction
  - » Opcode = 000 ~ 110
    - ■ I=0 : 0xxx ~ 6xxx, I=1: 8xxx

I=0 : Direct,
I=1 : Indirect

15 14 12 11 0

I	Opcode	Address

- Register-reference instruction
  - » 7xxx (7800 ~ 7001) : CLA, C

15 14 12 11 0

0	1	1	1	Register Operation

- Input-Output instruction
  - » Fxxx(F800 ~ F040) : INP, OU

15 14 12 11 0

1	1	1	1	I/O Operation

Computer System Architecture Chap. 5 Basic

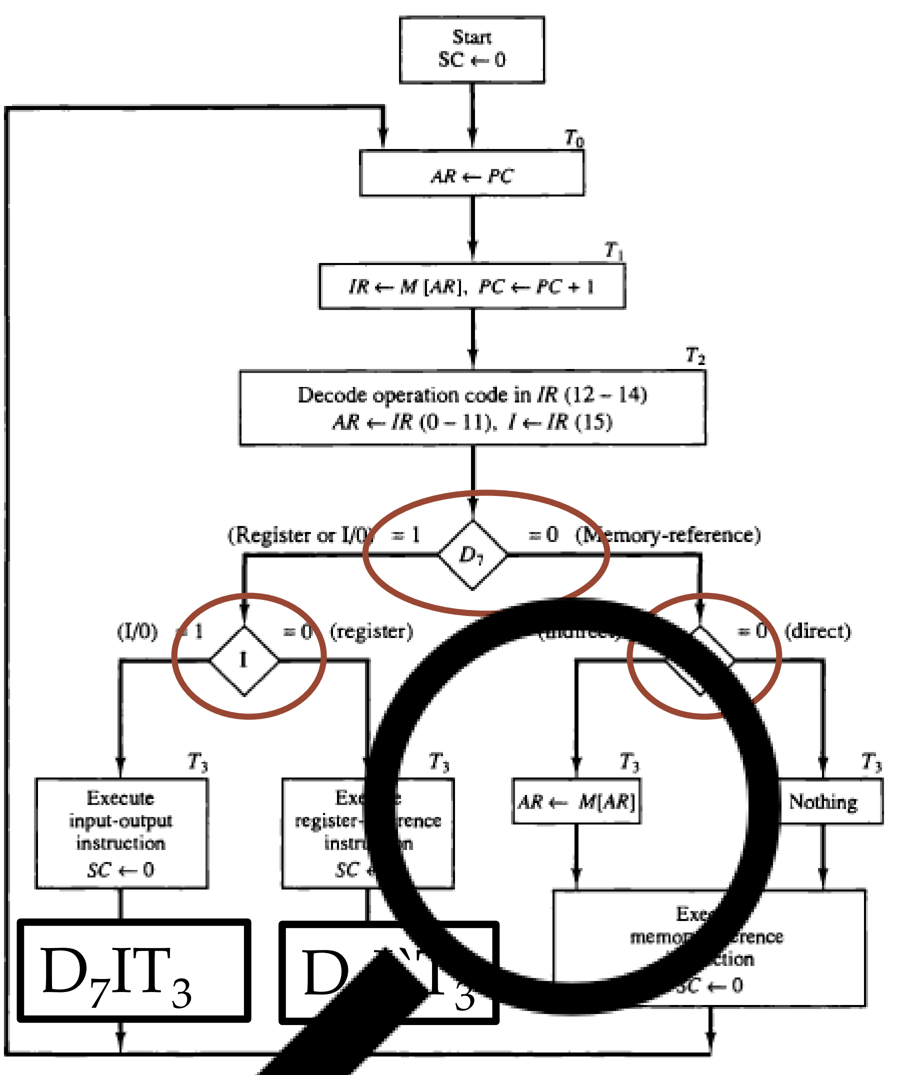
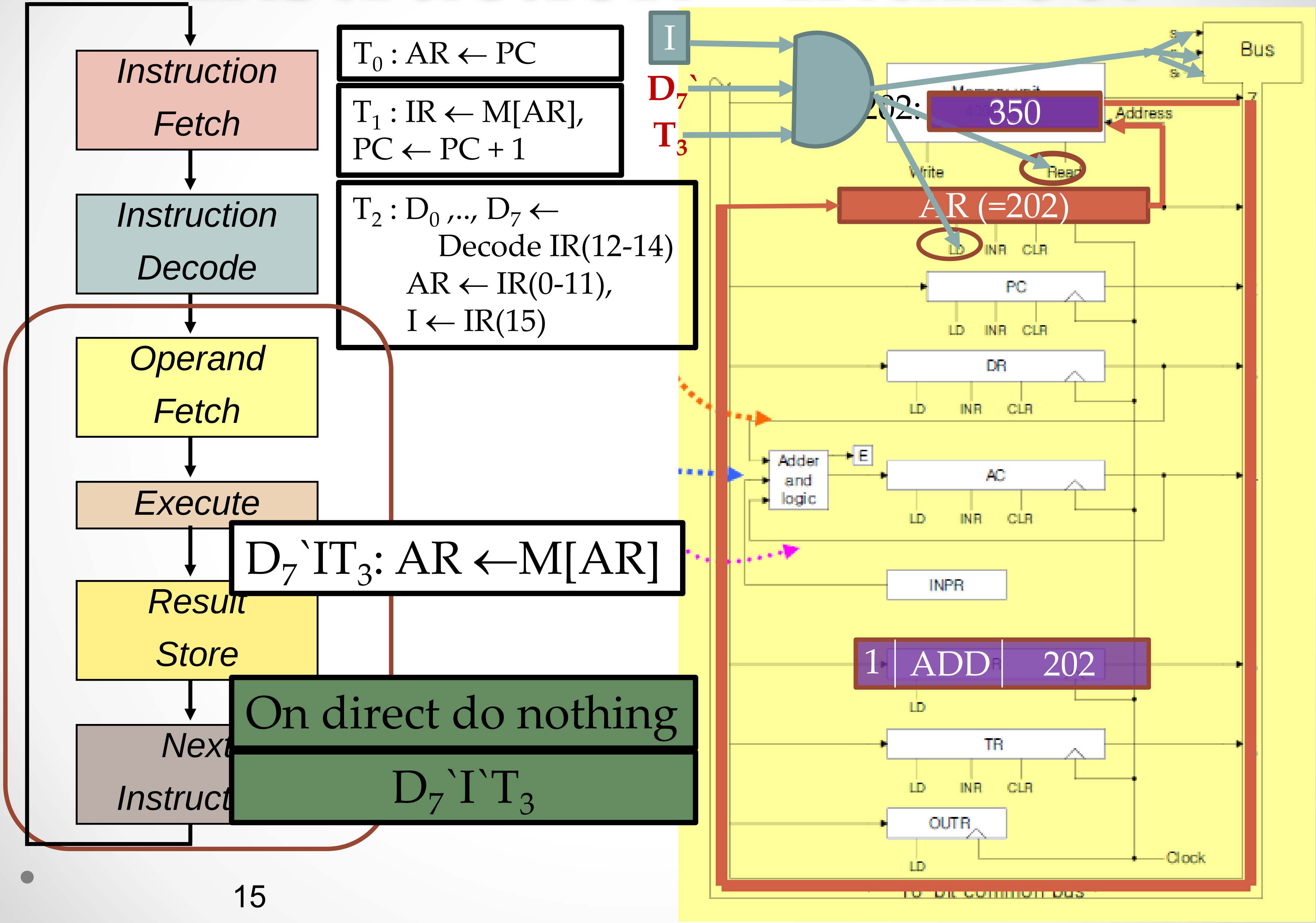


Figure 5-9 rt for instruction cycle (initial configuration).

Instruction - Indirect



$T_0 : AR \leftarrow PC$

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

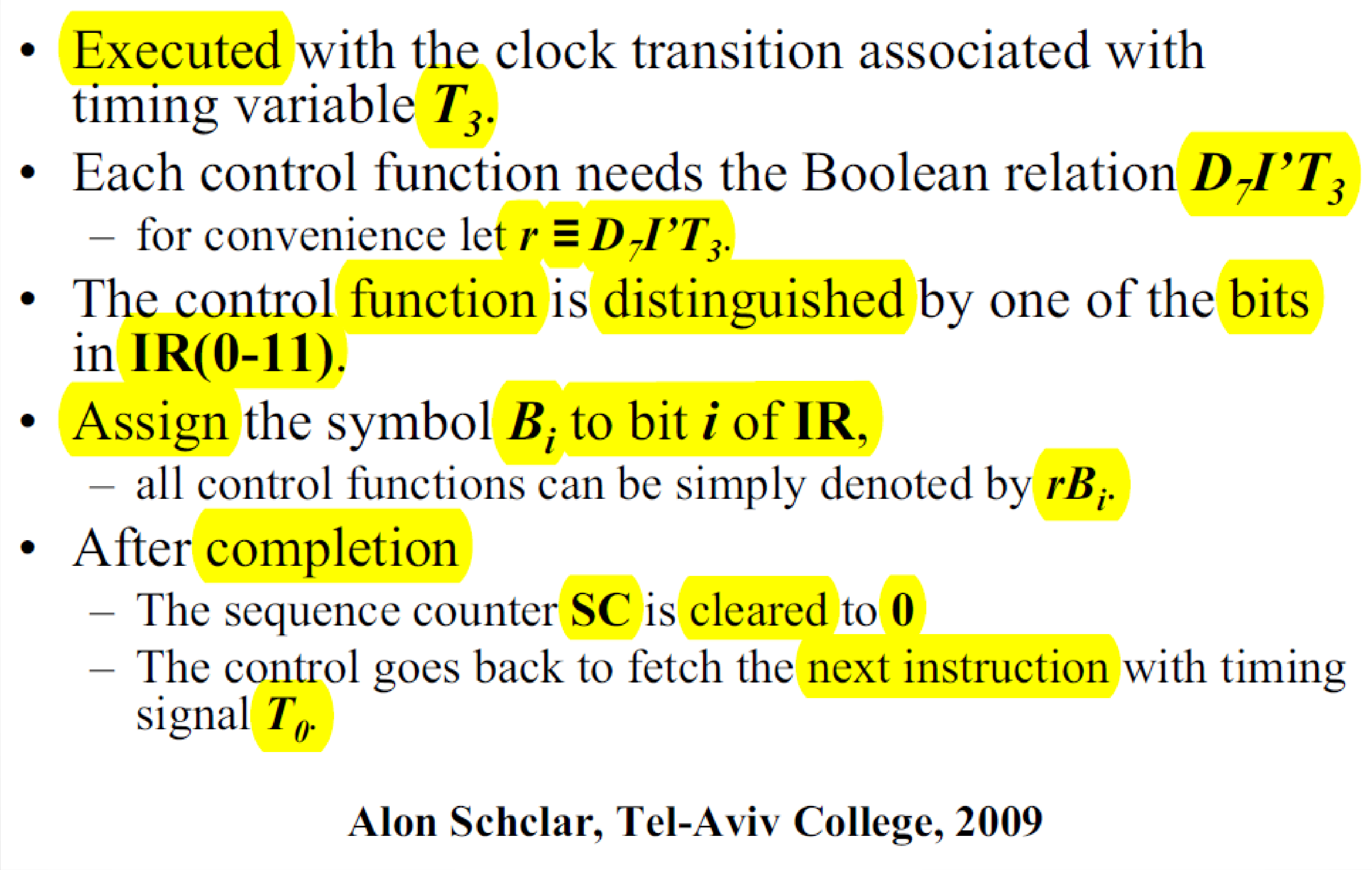
$T_2 : D_0, .., D_7 \leftarrow$ Decode IR(12-14)
$AR \leftarrow IR(0\text{-}11)$,
$I \leftarrow IR(15)$

$D_7`IT_3$: $AR \leftarrow M[AR]$

On direct do nothing

$D_7`I`T_3$

Register Reference Instructions



- Executed with the clock transition associated with timing variable T_3.
- Each control function needs the Boolean relation $D_7I'T_3$
  - for convenience let $r \equiv D_7I'T_3$.
- The control function is distinguished by one of the bits in **IR(0-11)**.
- Assign the symbol B_i to bit i of **IR**,
  - all control functions can be simply denoted by rB_i.
- After completion
  - The sequence counter **SC** is cleared to **0**
  - The control goes back to fetch the next instruction with timing signal T_0.

Alon Schclar, Tel-Aviv College, 2009

Register Reference Instructions

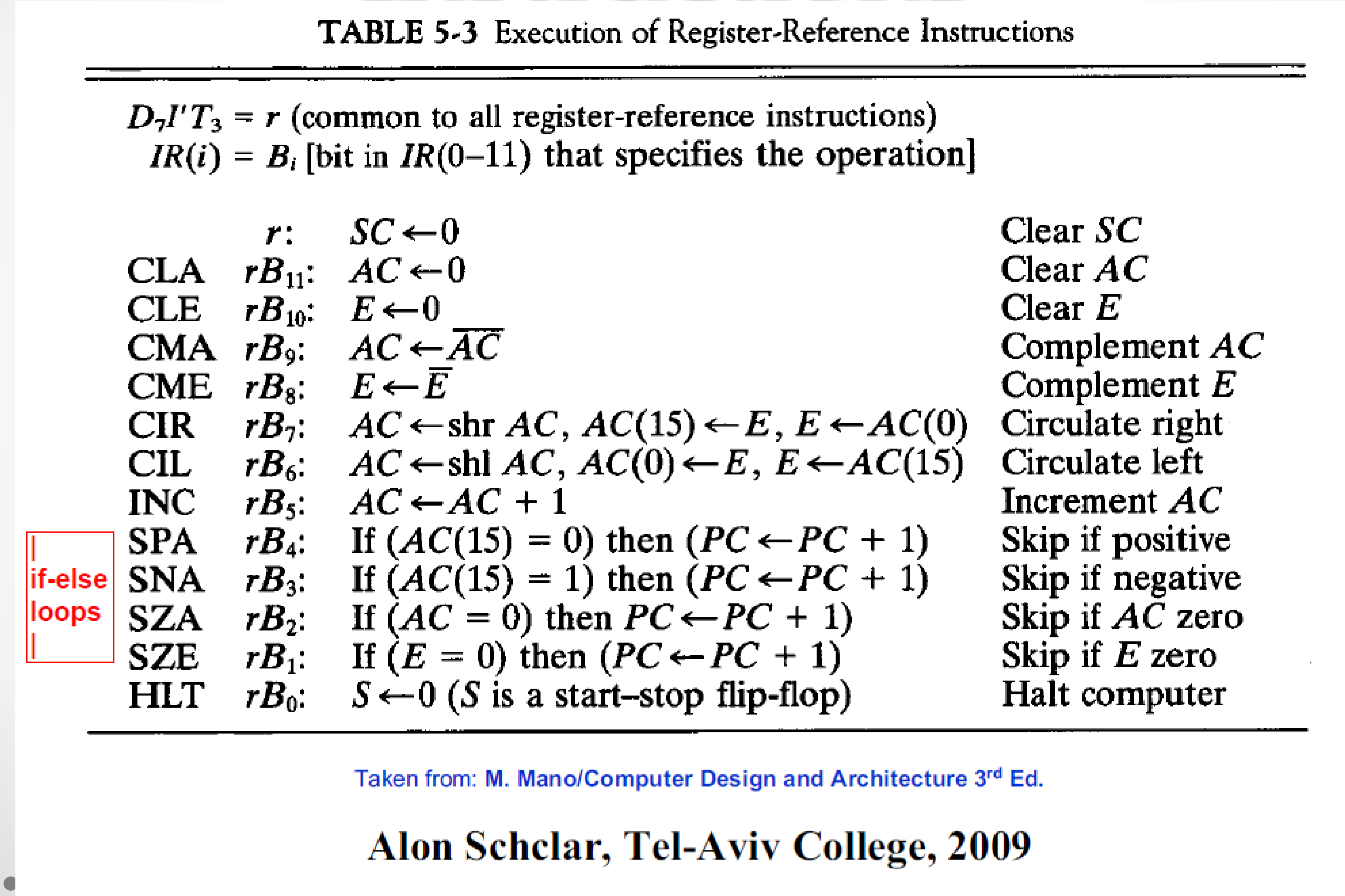


TABLE 5-3 Execution of Register-Reference Instructions

$D_7I'T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0\text{–}11)$ that specifies the operation]

	r:	$SC \leftarrow 0$	Clear SC
CLA	rB_{11}:	$AC \leftarrow 0$	Clear AC
CLE	rB_{10}:	$E \leftarrow 0$	Clear E
CMA	rB_9:	$AC \leftarrow \overline{AC}$	Complement AC
CME	rB_8:	$E \leftarrow \overline{E}$	Complement E
CIR	rB_7:	$AC \leftarrow \text{shr } AC,\ AC(15) \leftarrow E,\ E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow \text{shl } AC,\ AC(0) \leftarrow E,\ E \leftarrow AC(15)$	Circulate left
INC	rB_5:	$AC \leftarrow AC + 1$	Increment AC
SPA	rB_4:	If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$	Skip if positive
SNA	rB_3:	If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$	Skip if negative
SZA	rB_2:	If $(AC = 0)$ then $PC \leftarrow PC + 1)$	Skip if AC zero
SZE	rB_1:	If $(E = 0)$ then $(PC \leftarrow PC + 1)$	Skip if E zero
HLT	rB_0:	$S \leftarrow 0$ (S is a start–stop flip-flop)	Halt computer

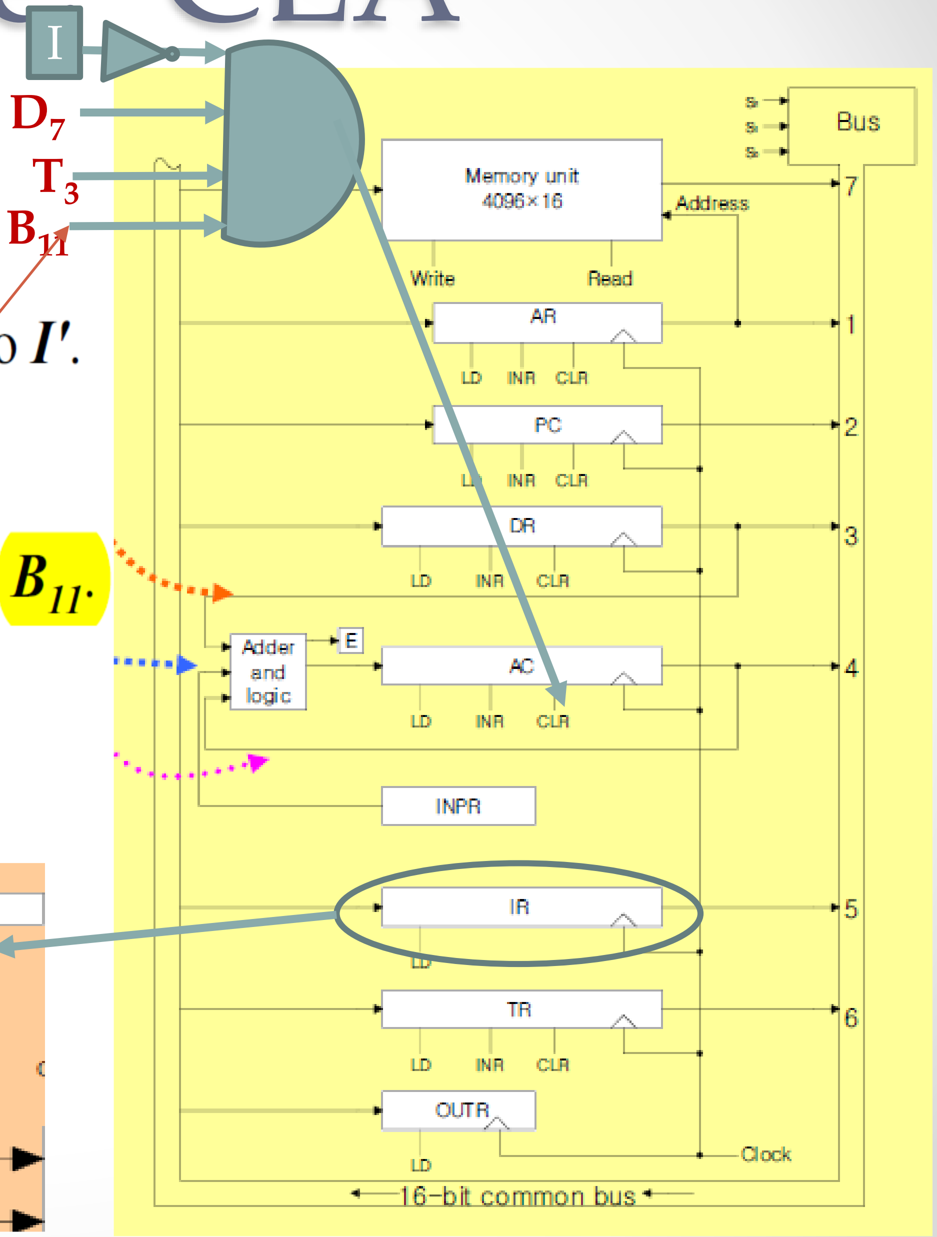
|
if-else
loops
|

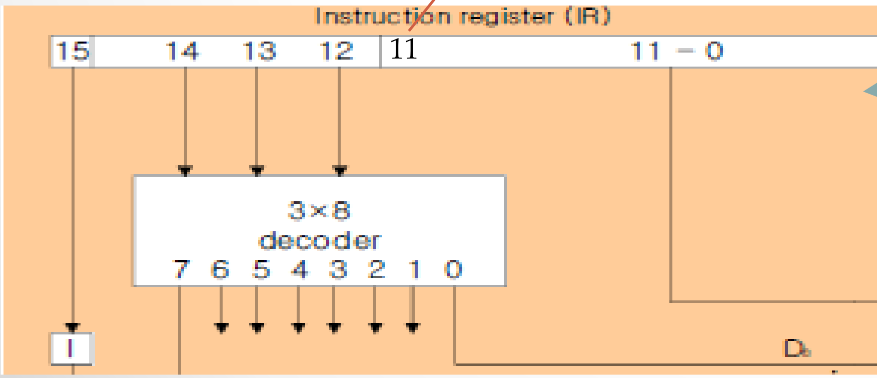
Taken from: M. Mano/Computer Design and Architecture 3rd Ed.

Alon Schclar, Tel-Aviv College, 2009

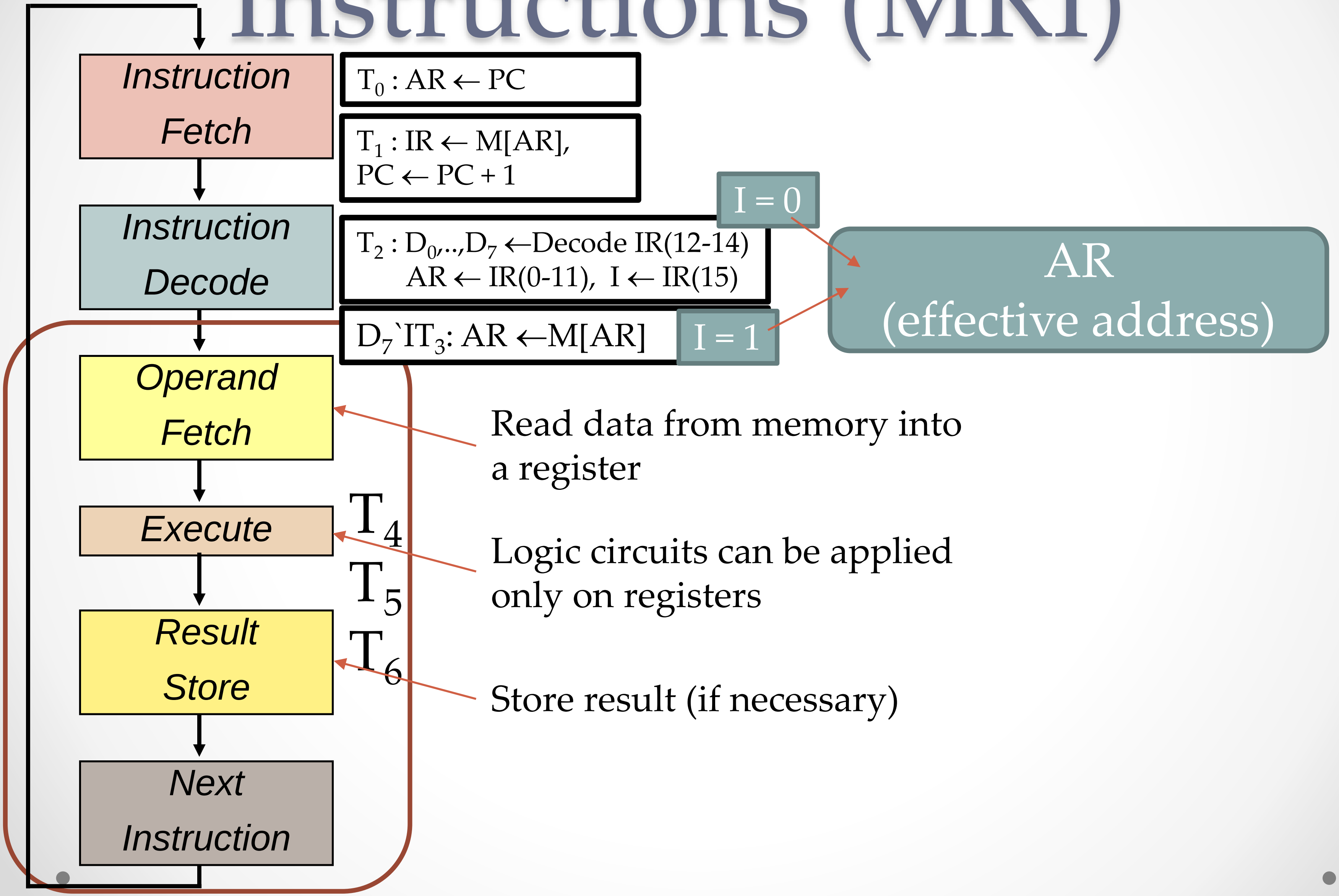
Example: CLA

- Hexadecimal code $(7800)_{16} = (0111\ 1000\ 0000\ 0000)_2$.
- The first bit is a zero and is equivalent to I'.
- Next three bits are the opcode and are recognized from decoder output D_7.
- Bit *11* in IR is *1* and is recognized from B_{11}.
- The control function that initiates the microoperation for this instruction is $D_7I'T_3B_{11} = rB_{11}$.





Memory Reference Instructions (MRI)



Instruction Fetch

Instruction Decode

Operand Fetch

Execute

Result Store

Next Instruction

T_0 : AR ← PC

T_1 : IR ← M[AR],
PC ← PC + 1

T_2 : D_0,..,D_7 ←Decode IR(12-14)
AR ← IR(0-11), I ← IR(15)

D_7`IT_3: AR ←M[AR]

I = 0

I = 1

AR
(effective address)

T_4

T_5

T_6

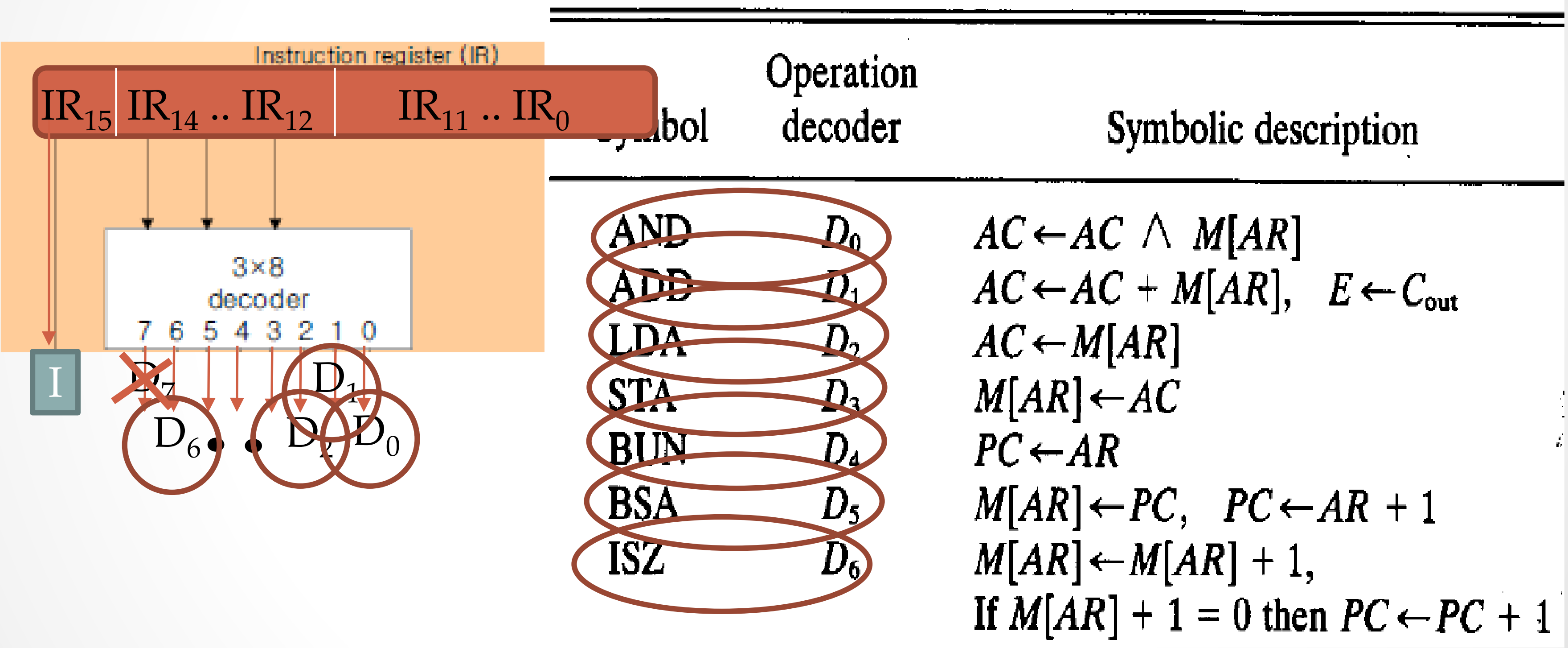
Read data from memory into a register

Logic circuits can be applied only on registers

Store result (if necessary)

Memory Reference Instructions (MRI)

At the end of the T_3 cycle, AR holds the effective Address



Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], \quad E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$, If $M[AR] + 1 = 0$ then $PC \leftarrow PC + 1$

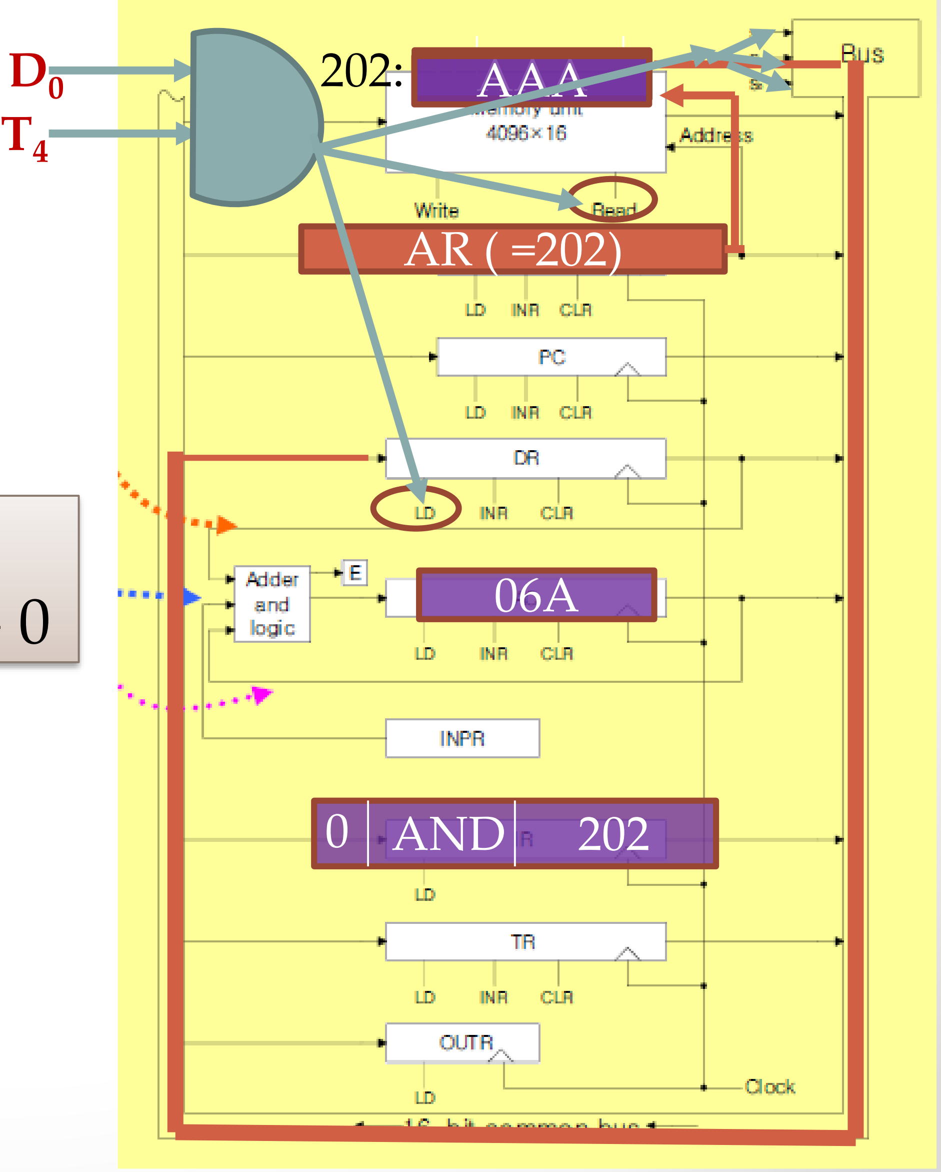
MRI - AND

At the end of the T_3 cycle, AR holds the **effective Address**

$AND : AC \leftarrow AC \wedge M[AR]$

$D_0T_4 : \; DR \leftarrow M[AR]$

$D_0T_5 : \; AC \leftarrow AC \wedge DR, \; SC \leftarrow 0$



●

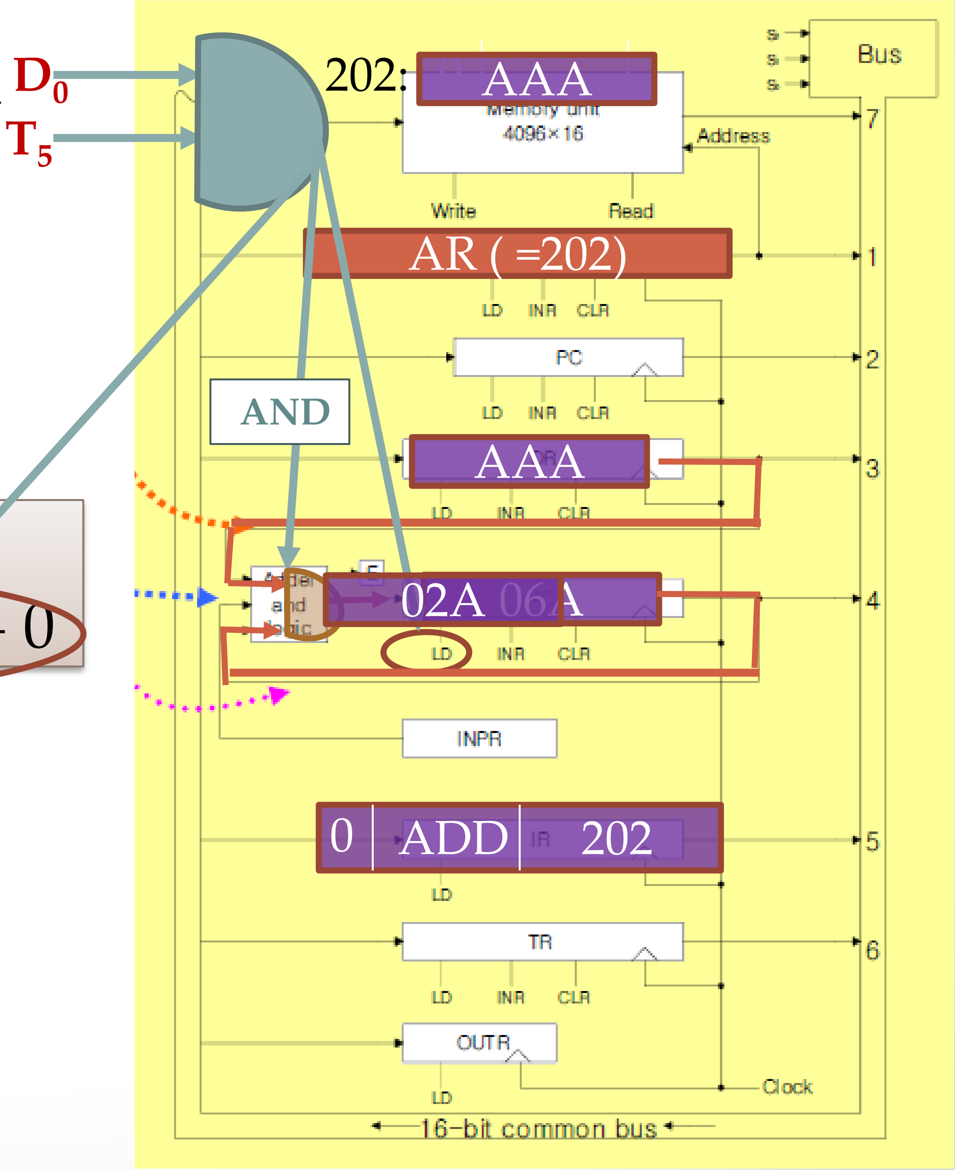
MRI - AND

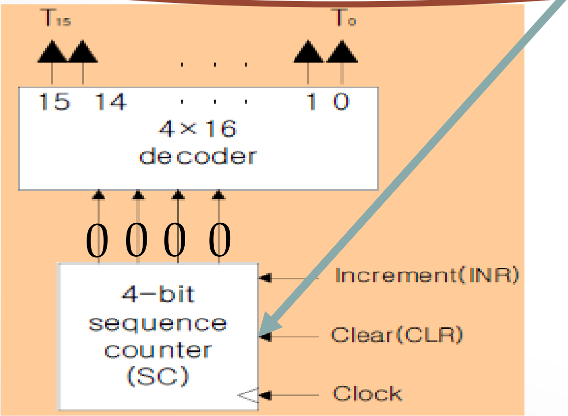
At the end of the T_3 cycle, AR holds the **effective Address**

AND : AC ← AC ∧ M[AR]

D_0T_4 : DR ← M[AR]

D_0T_5 : AC ← AC ∧ DR, SC ← 0



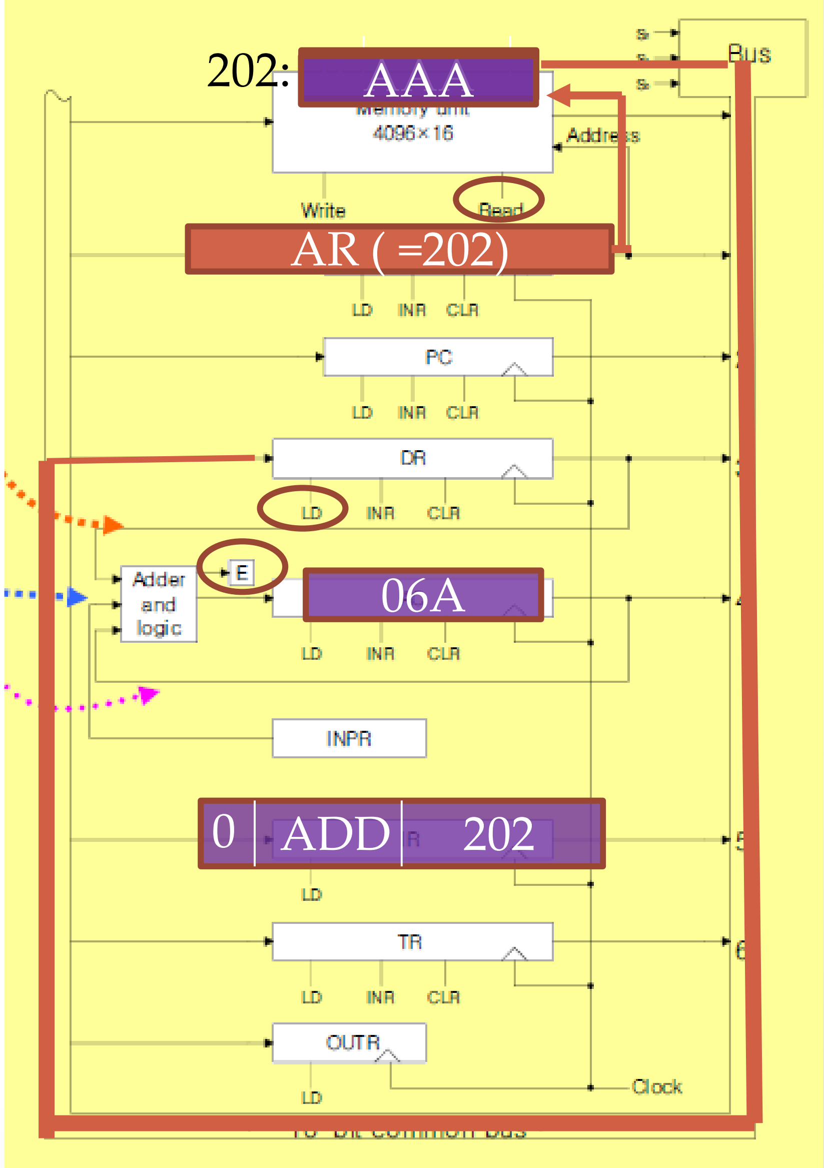


MRI - ADD

At the end of the T_3 cycle, AR holds the **effective Address**

$$\begin{aligned} \text{ADD} : AC &\leftarrow AC + M[AR], \\ E &\leftarrow C_{out} \end{aligned}$$

$$\begin{aligned} D_1T_4 &: \; DR \leftarrow M[AR] \\ D_1T_5 &: \; AC \leftarrow AC + DR, \\ & \quad E \leftarrow C_{out}, \; SC \leftarrow 0 \end{aligned}$$



MRI - LDA

At the end of the T_3 cycle, AR holds the **effective Address**

$$LDA : AC \leftarrow M[AR]$$

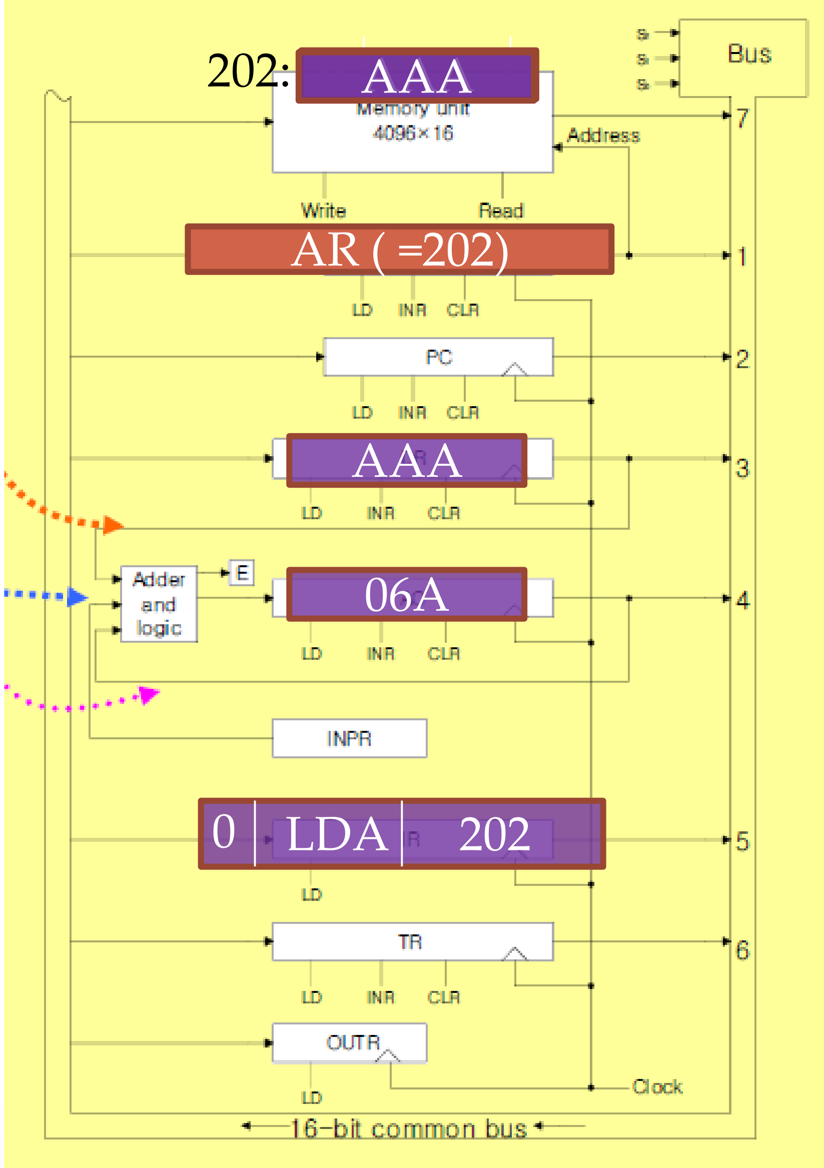
$$D_2T_4 : \ DR \leftarrow M[AR]$$
$$D_2T_5 : \ AC \leftarrow DR, SC \leftarrow 0$$

Why not connecting the bus to the inputs of ***AC*** ?

- a delay is encountered in the adder and logic circuit.

  Time(Mem read) + Time(Bus transfer) + Time(A&L) > 1 cycle

- Not connecting the bus to the inputs of AC maintains **one clock cycle per microoperation**.

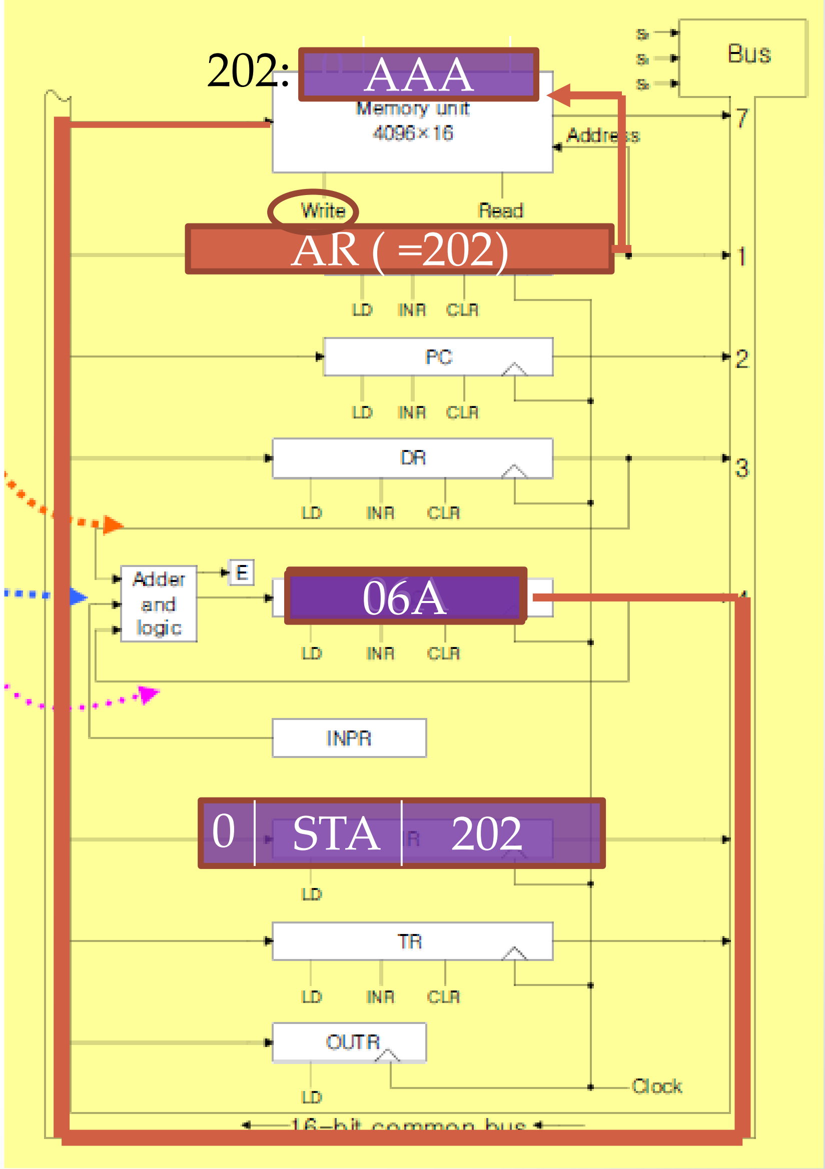


MRI - STA

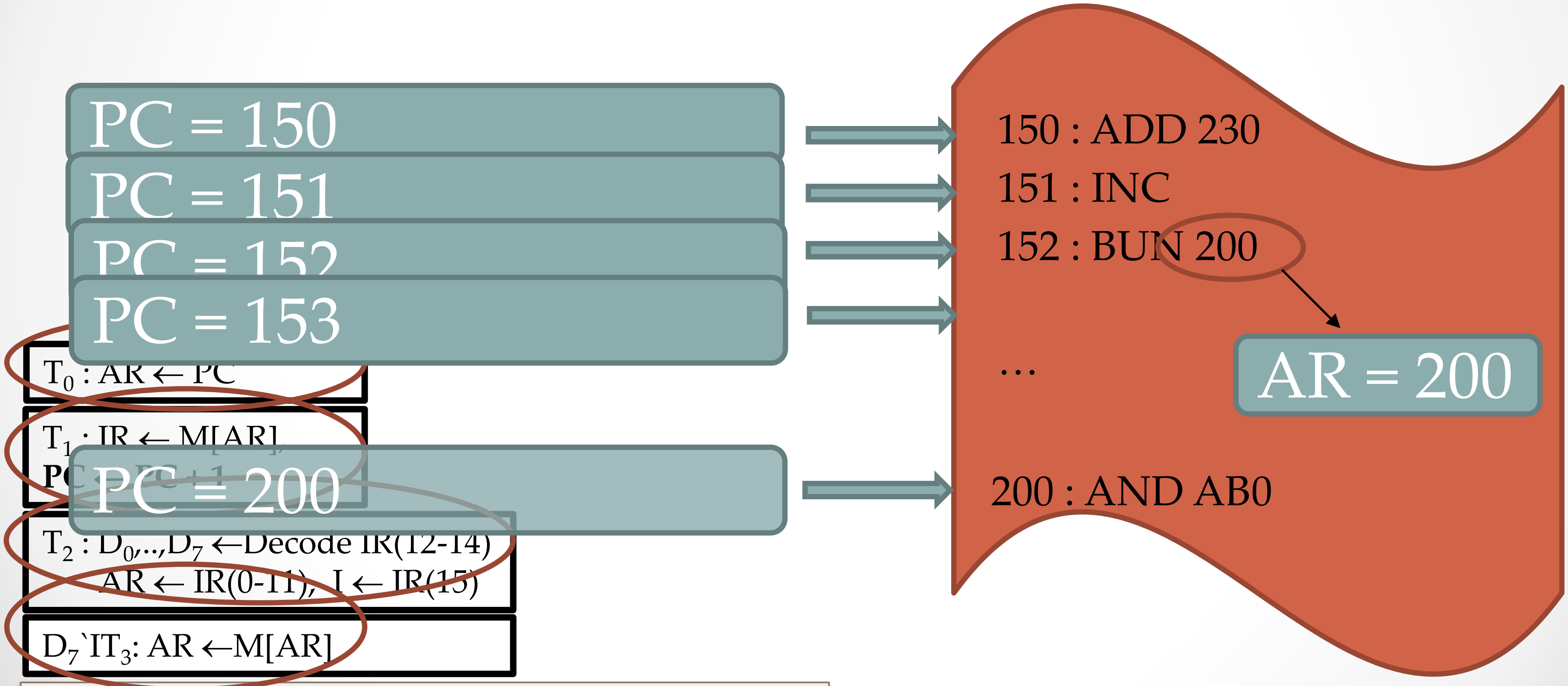
At the end of the T_3 cycle, AR holds the **effective Address**

$\text{STA} : M[AR] \leftarrow AC$

$D_3T_4 : M[AR] \leftarrow AC, SC \leftarrow 0$



MRI - BUN



T_0 : AR ← PC

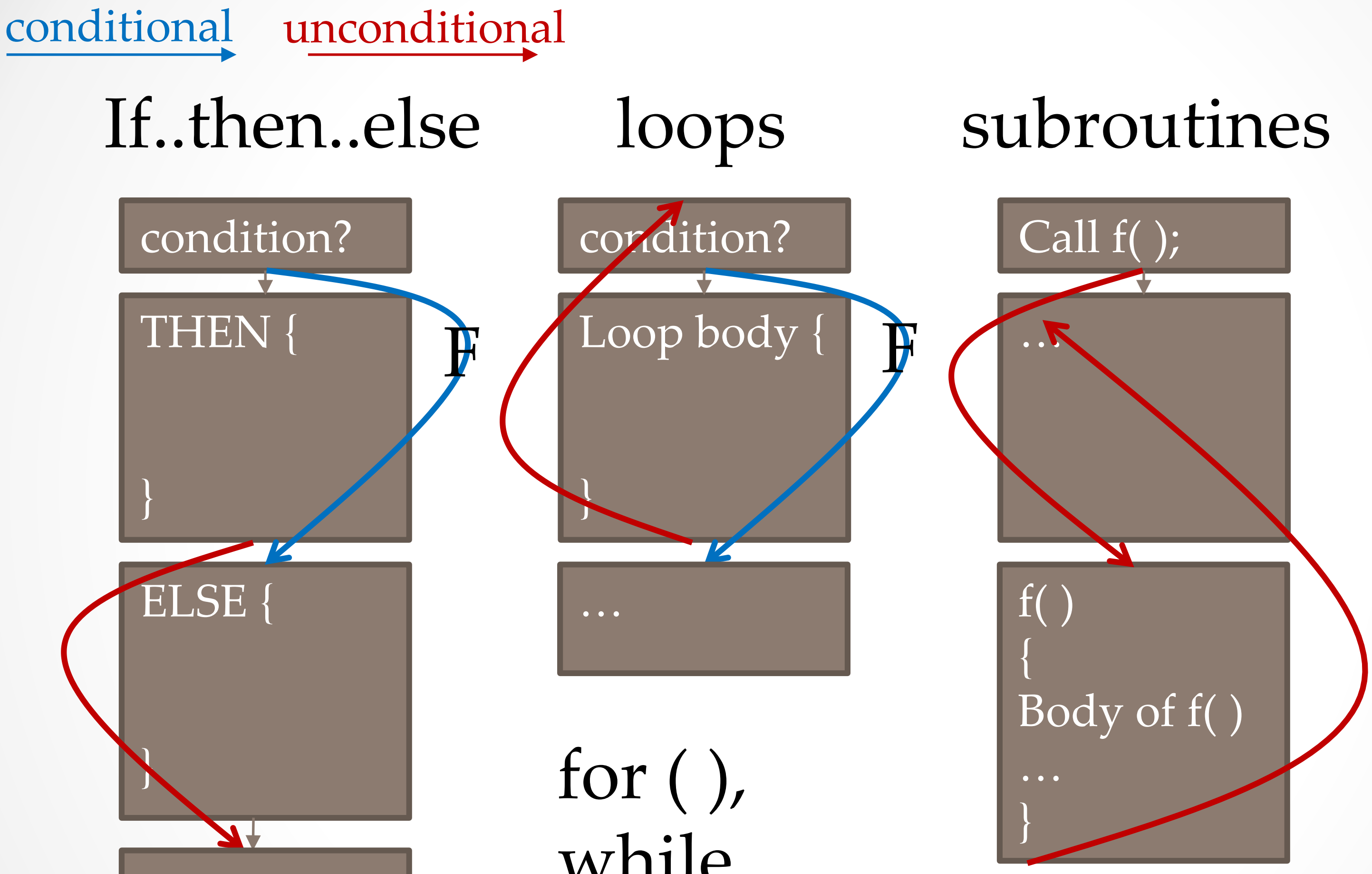
T_1 : IR ← M[AR], PC ← PC + 1

T_2 : $D_0,..,D_7$ ←Decode IR(12-14)
AR ← IR(0-11), I ← IR(15)

D_7`IT_3: AR ←M[AR]

D_4T_4 : PC ← AR, SC ← 0

Conditional & Unconditional Jumps



conditional

unconditional

If..then..else

condition?

THEN {

}

F

ELSE {

}

…

loops

condition?

Loop body {

}

F

…

subroutines

Call f( );

…

f( )

{

Body of f( )

…

}

for ( ),
while,
do .. while

goto

switch / case

MRI - BSA

Branch to subroutine and
save the return address

BSA : M[AR] ← PC, PC ← AR + 1

T_0 : AR ← PC

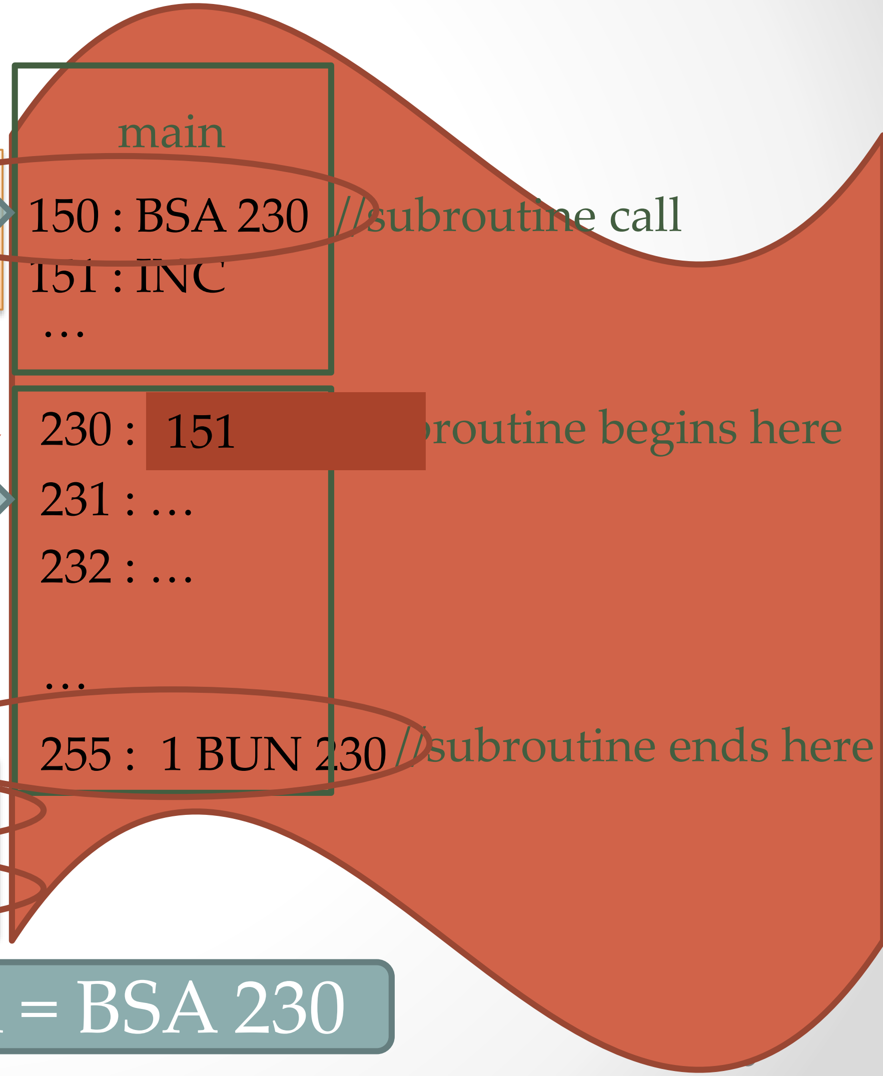
T_1 : IR ← M[AR],
PC ← PC + 1

T_2 : $D_0,..,D_7$ ←Decode IR(12-14)
AR ← IR(0-11), I ← IR(15)

D_7\`IT_3: AR ←M[AR]

D_5T_4 : M[AR] ← PC, AR ← AR + 1

D_5T_5 : PC ← AR, SC ← 0



main

150 : BSA 230 //subroutine call

151 : INC

…

230 : 151 broutine begins here

231 : …

232 : …

…

255 : 1 BUN 230 //subroutine ends here

PC = 231

AR = 231

IR = BSA 230

MRI – BUN (cont. BSA)

Branch to subroutine and
save the return address

BSA : M[AR] ← PC, PC ← AR + 1

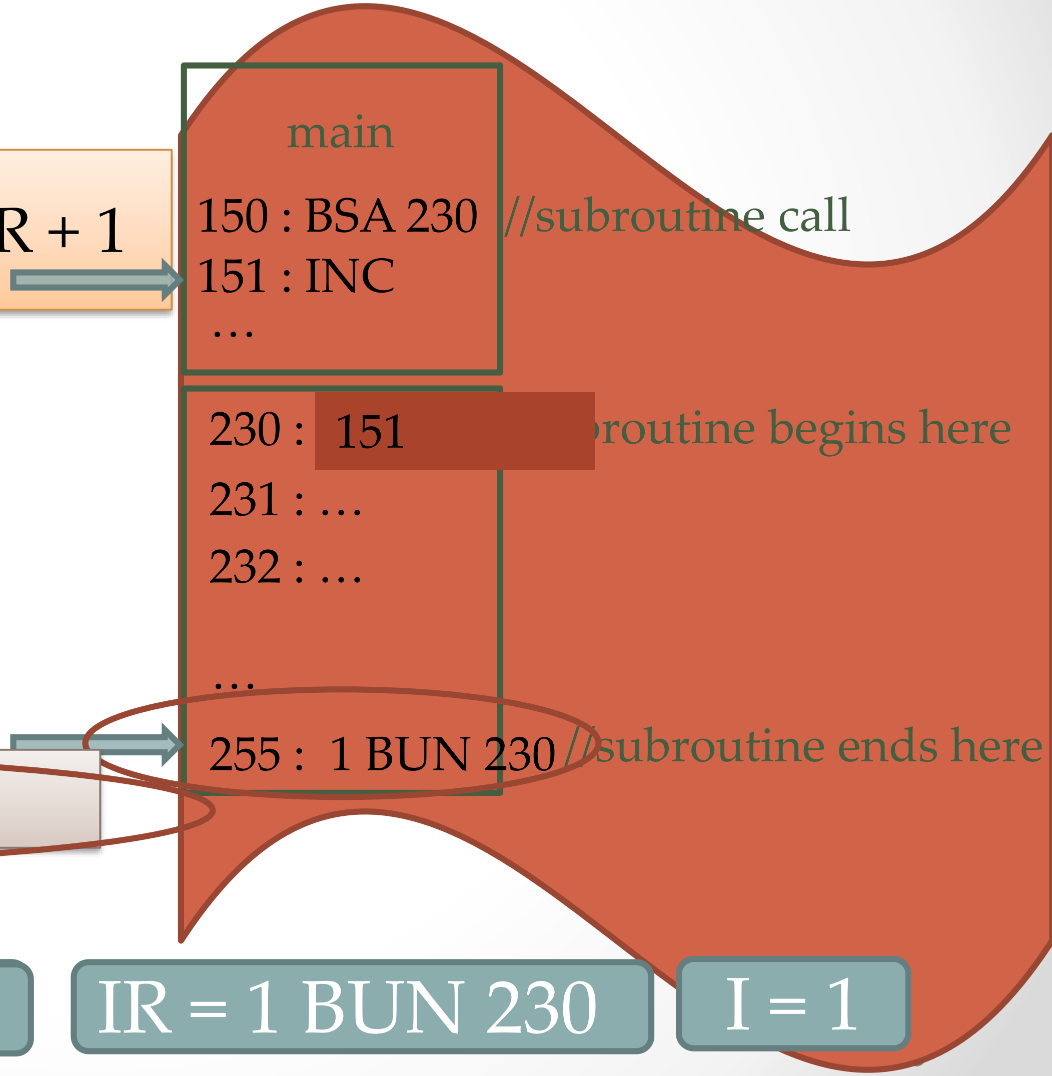
T_0 : AR ← PC

T_1 : IR ← M[AR],
PC ← PC + 1

T_2 : $D_0,..,D_7$ ←Decode IR(12-14)
AR ← IR(0-11), I ← IR(15)

D_7`IT_3: AR ←M[AR]

D_4T_4 : PC ← AR, SC ← 0



main

150 : BSA 230 //subroutine call
151 : INC
…

230 : 151 broutine begins here
231 : …
232 : …

…

255 : 1 BUN 230 //subroutine ends here

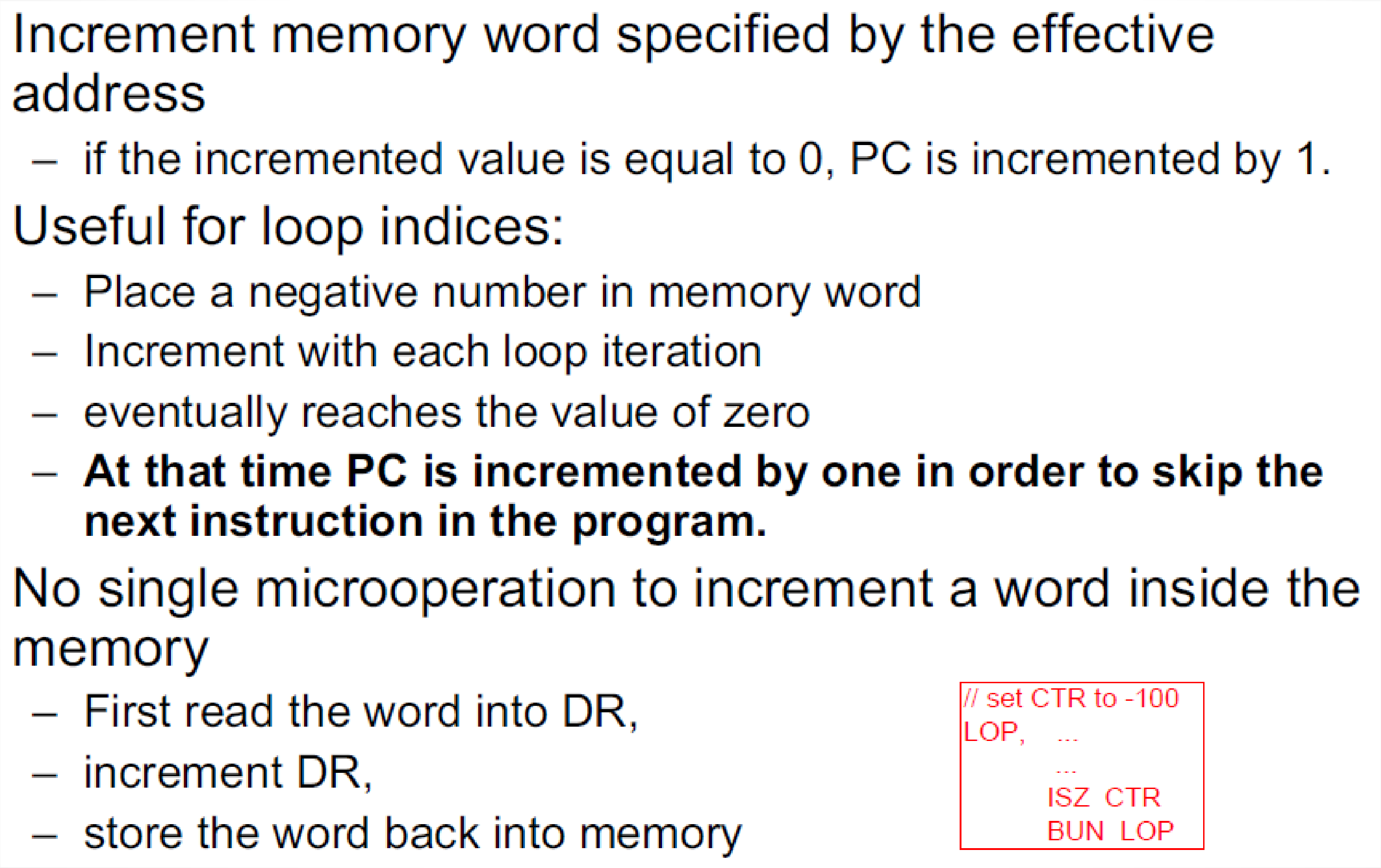
PC = 151

AR = 151

IR = 1 BUN 230

I = 1

MRI - ISZ



Increment memory word specified by the effective address

- if the incremented value is equal to 0, PC is incremented by 1.

Useful for loop indices:

- Place a negative number in memory word
- Increment with each loop iteration
- eventually reaches the value of zero
- **At that time PC is incremented by one in order to skip the next instruction in the program.**

No single microoperation to increment a word inside the memory

- First read the word into DR,
- increment DR,
- store the word back into memory

```
// set CTR to -100
LOP,    ...
        ...
        ISZ  CTR
        BUN  LOP
```

MRI - ISZ

Increment memory word and skip next instruction if memory word equals to zero.

D_6T_4 : DR ← M[AR]

D_6T_5 : DR ← DR + 1

D_6T_6 : M[AR] ← DR,

if (DR = 0) then PC ← PC + 1,

SC ← 0

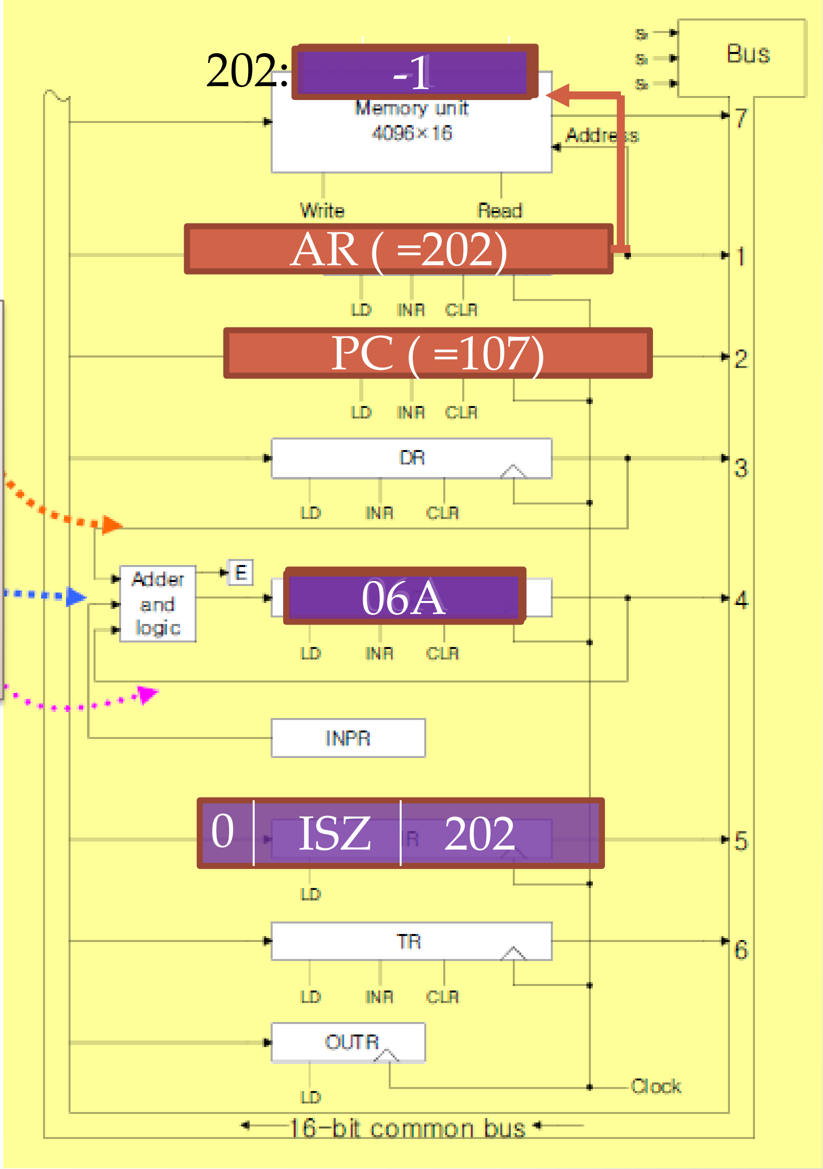
….

106 : ISZ 202

107 : BUN 150

108 : BUN 090

….



MRI - ISZ

Increment memory word and skip next instruction if memory word equals to zero.

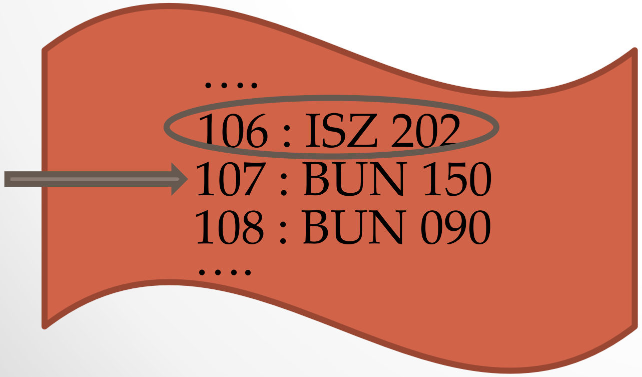
$D_6T_4 : DR \leftarrow M[AR]$

$D_6T_5 : DR \leftarrow DR + 1$

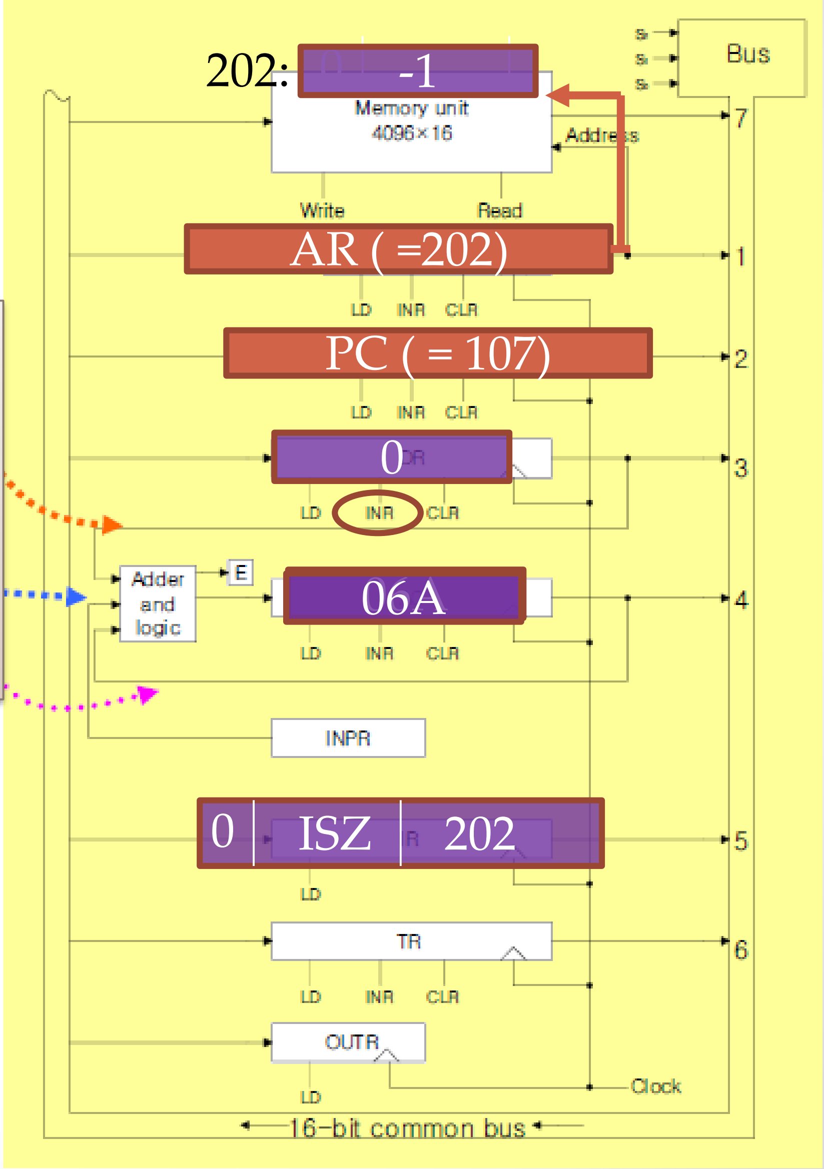
$D_6T_6 : M[AR] \leftarrow DR,$

$\quad \text{if } (DR = 0) \text{ then } PC \leftarrow PC + 1,$

$\quad SC \leftarrow 0$



….
106 : ISZ 202
107 : BUN 150
108 : BUN 090
….



MRI - ISZ

Increment memory word and skip next instruction if memory word equals to zero.

$D_6T_4 : DR \leftarrow M[AR]$

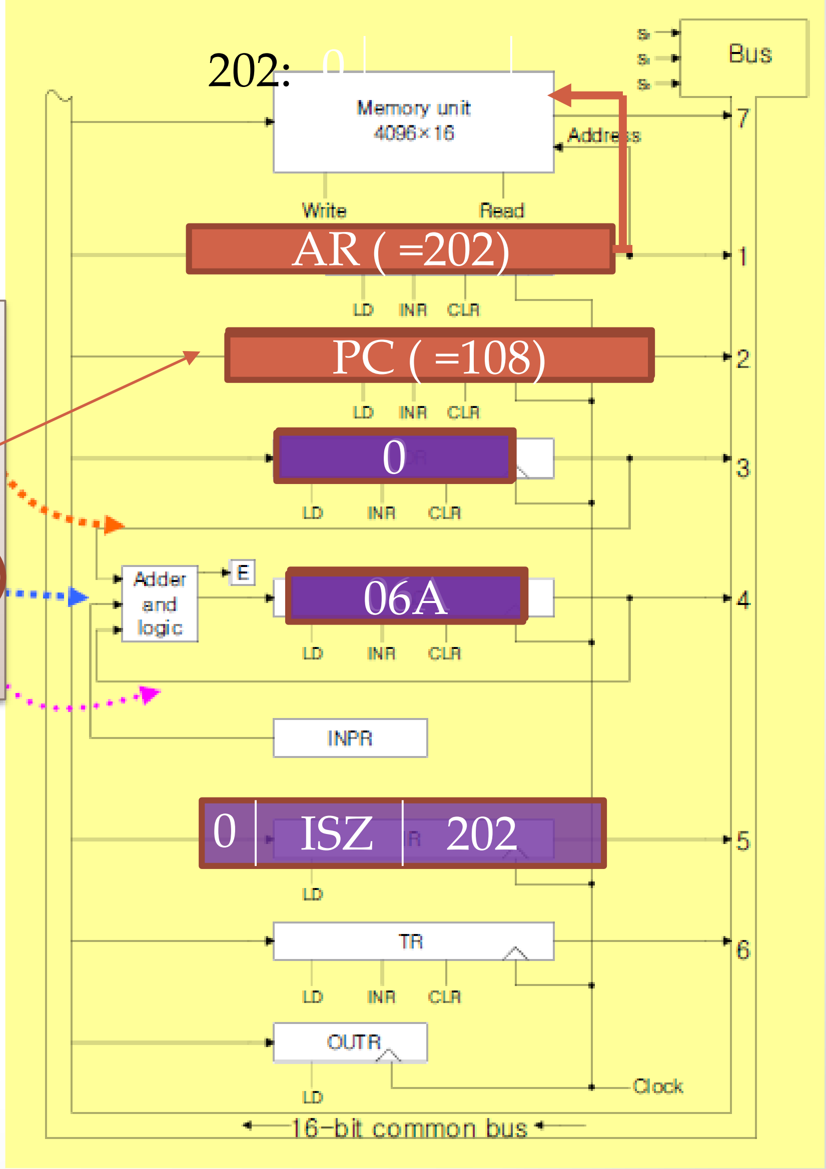
$D_6T_5 : DR \leftarrow DR + 1$

$D_6T_6 : M[AR] \leftarrow DR,$

$\quad$ if $(DR = 0)$ then $PC \leftarrow PC + 1,$

$\quad SC \leftarrow 0$

....
106 : ISZ 202
107 : BUN 150
108 : BUN 090
....



MRI – ISZ (another example)

Increment memory word and skip next instruction if memory word equals to zero.

$D_6T_4 : DR \leftarrow M[AR]$

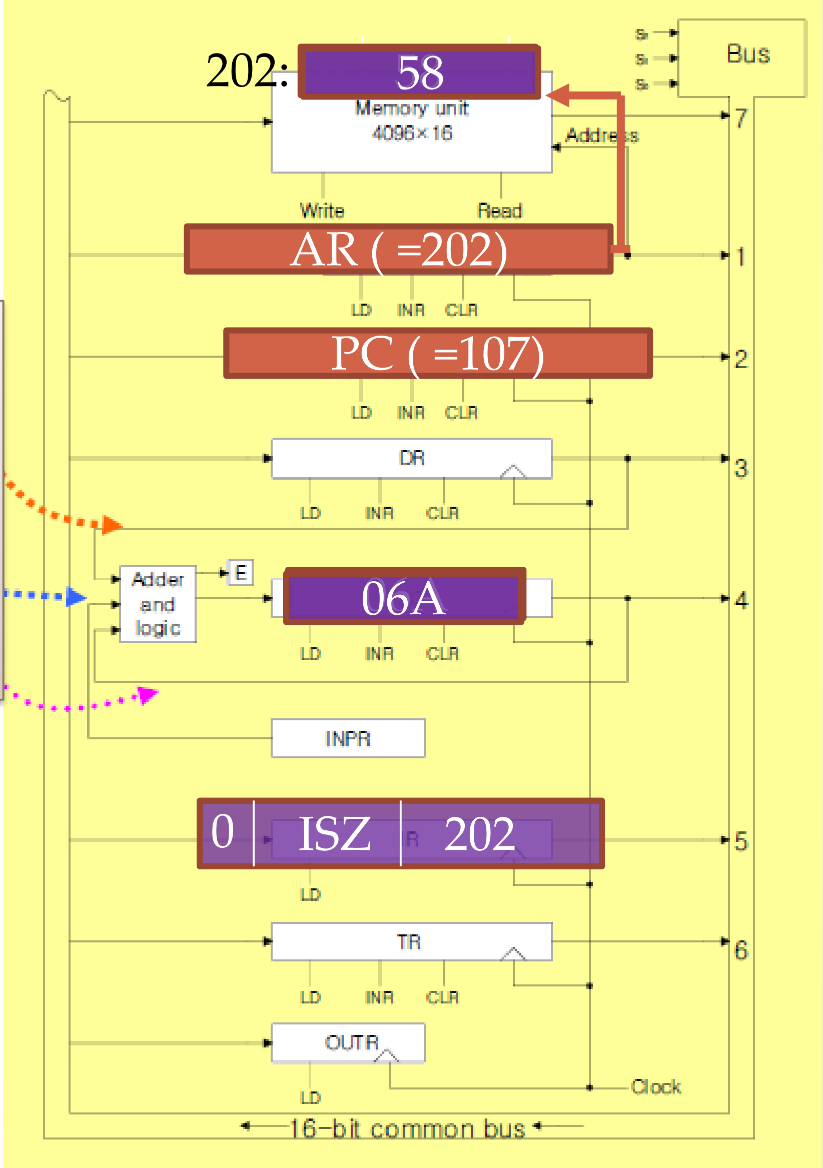
$D_6T_5 : DR \leftarrow DR + 1$

$D_6T_6 : M[AR] \leftarrow DR,$

$\quad \text{if } (DR = 0) \text{ then } PC \leftarrow PC + 1,$

$\quad SC \leftarrow 0$

….
106 : ISZ 202
107 : BUN 150
108 : BUN 090
….



MRI - ISZ

Increment memory word and skip next instruction if memory word equals to zero.

$D_6T_4 : DR \leftarrow M[AR]$

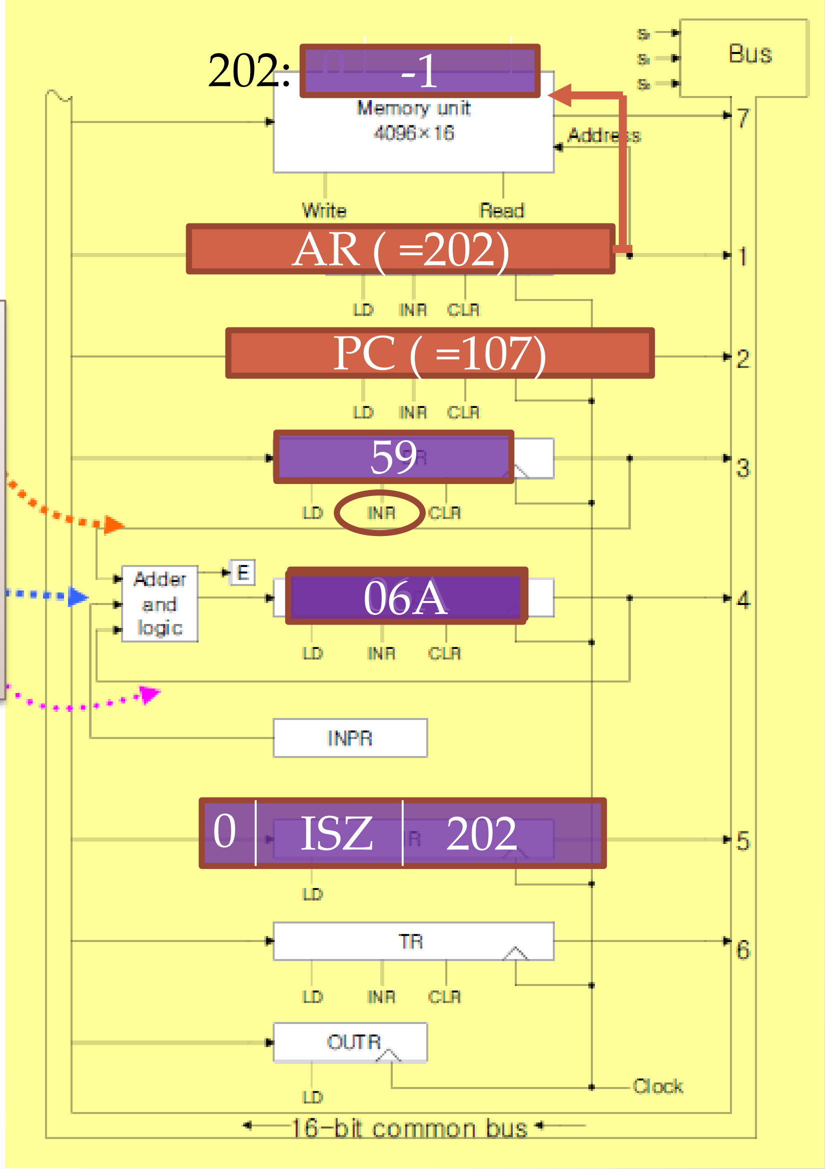
$D_6T_5 : DR \leftarrow DR + 1$

$D_6T_6 : M[AR] \leftarrow DR,$

$\quad$ if $(DR = 0)$ then $PC \leftarrow PC + 1,$

$\quad SC \leftarrow 0$

….
106 : ISZ 202
107 : BUN 150
108 : BUN 090
….



MRI - ISZ

Increment memory word and skip next instruction if memory word equals to zero.

$D_6T_4 : DR \leftarrow M[AR]$

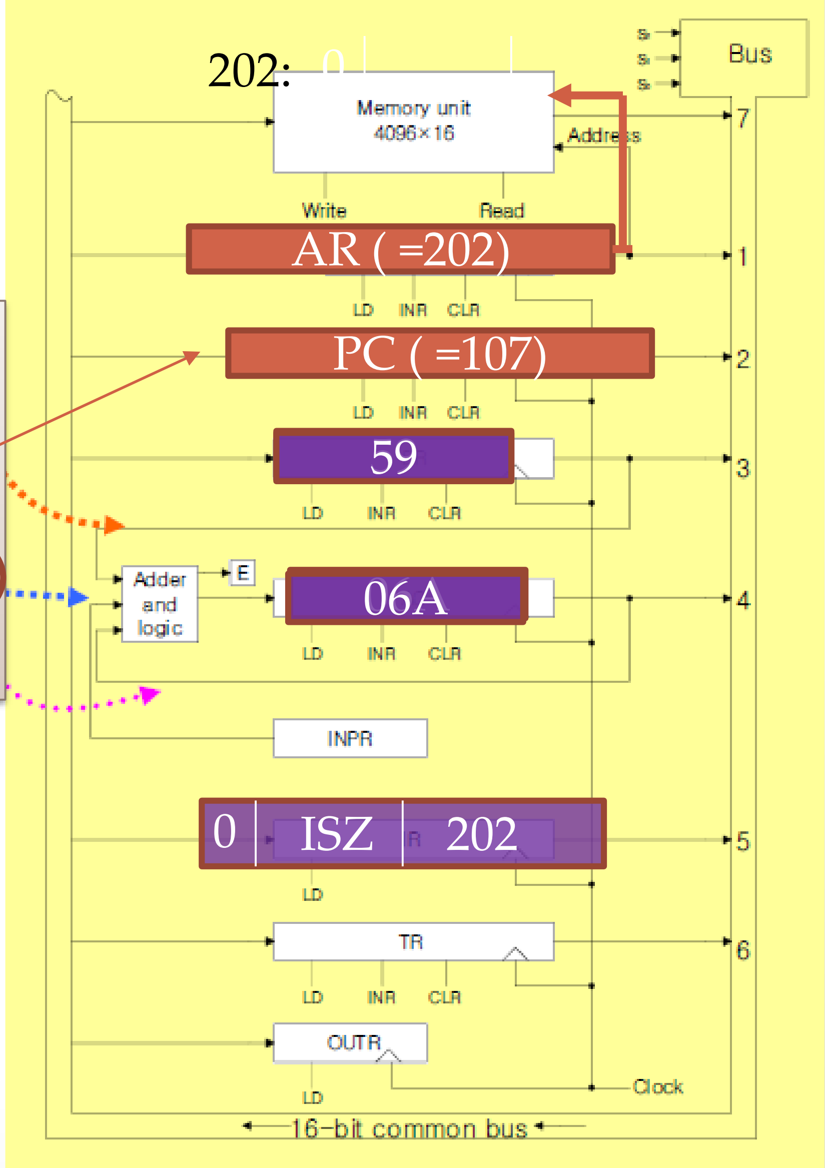
$D_6T_5 : DR \leftarrow DR + 1$

$D_6T_6 : M[AR] \leftarrow DR,$

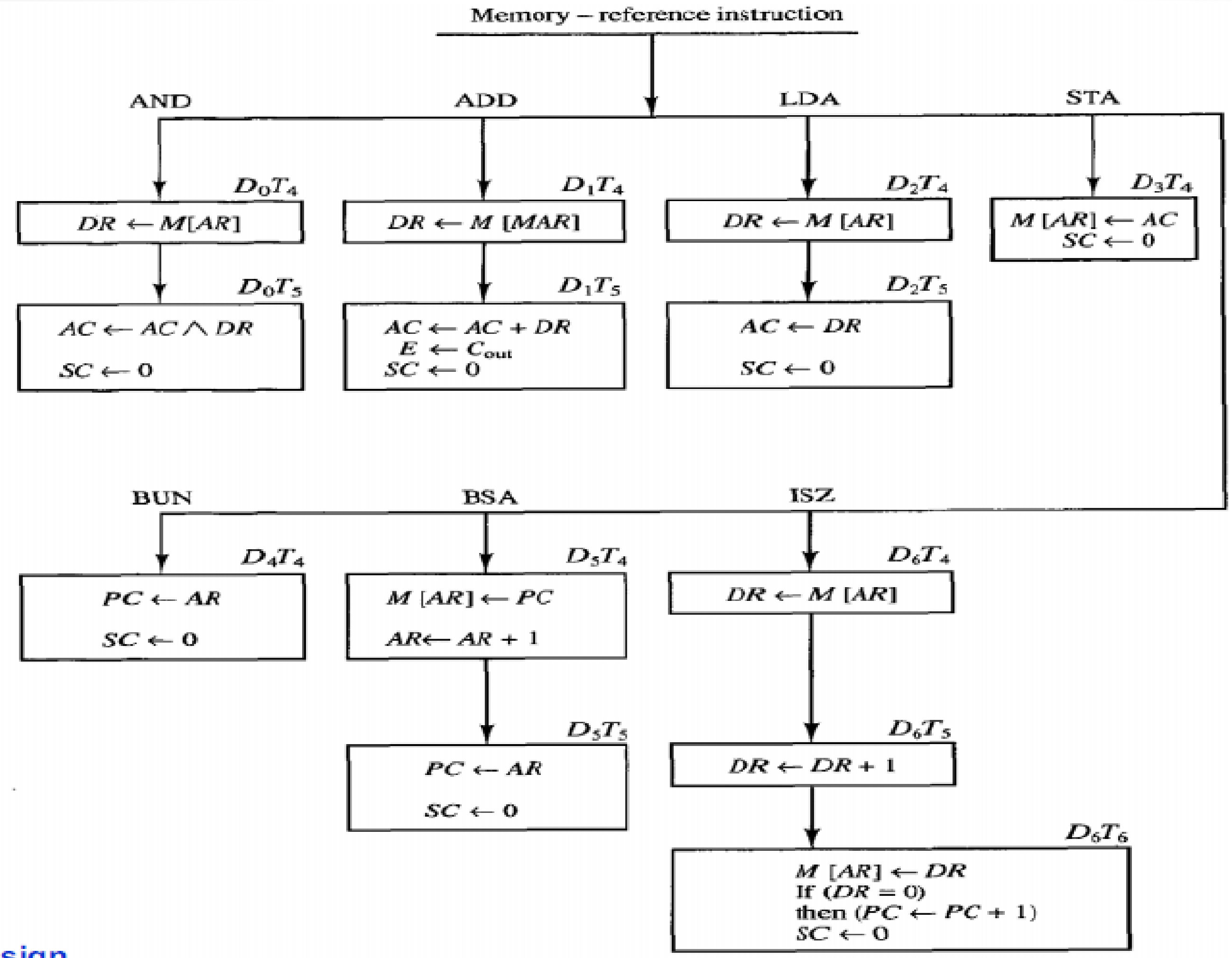
$\quad \text{if } (DR = 0) \text{ then } PC \leftarrow PC + 1,$

$\quad SC \leftarrow 0$

....
106 : ISZ 202
107 : BUN 150
108 : BUN 090
....



Memory Reference Flowchart



Memory – reference instruction

AND

D_0T_4

$DR \leftarrow M[AR]$

D_0T_5

$AC \leftarrow AC \wedge DR$

$SC \leftarrow 0$

ADD

D_1T_4

$DR \leftarrow M\ [MAR]$

D_1T_5

$AC \leftarrow AC + DR$

$E \leftarrow C_{out}$

$SC \leftarrow 0$

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 $(PC \leftarrow PC + 1)$

$SC \leftarrow 0$

esign

rd Ed.

Figure 5-11 Flowchart for memory-reference instructions.