Computer Architecture ECS-355

Mr. Abdul Rehman **email id: abdul@northern.edu.pk** **Whatsapp# 0308-7792217**

(Week 5) Lecture 9-10

Learning objectives:

- Review of the last lecture
- Determine different types of instructions
- Understand the concept of register reference instruction
- Understand how memory reference instructions are executed

(Note: It is important to note that we are considering "Basic computer architecture" processor which we were discussing in the class and it's the continuation of the same concepts. Another important point is that brief part of the last lecture is repeated. Audio clip will be sent separately for every diagram.)

Resources: Beside these lecture handouts, this lesson will draw from the following

Text Book: Computer System Architecture by Morris Mano (3rd Edition) and **Reference book:Computer Architecture, by William Stallings (4th Edition)**.

Lecture:

INSTRUCTION CYCLE

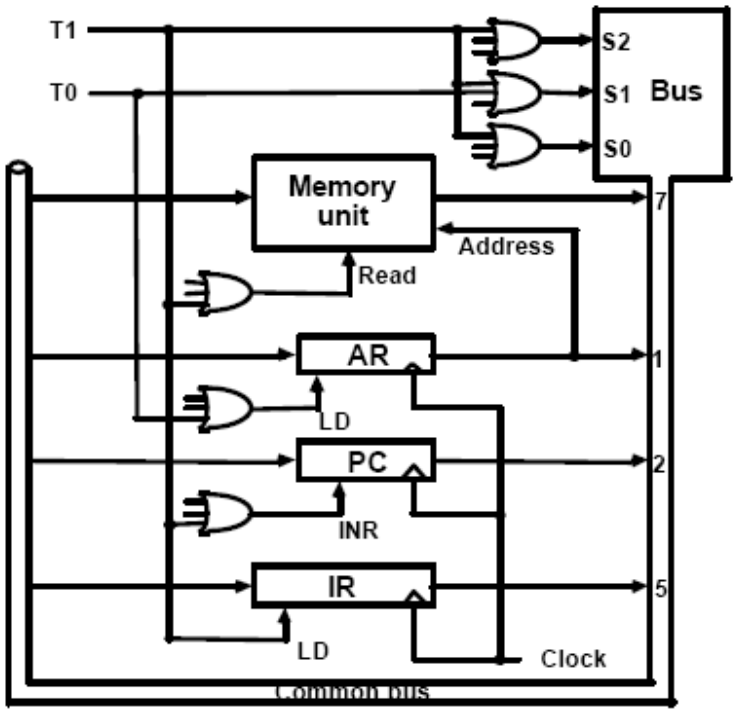
This section presents the steps an instruction needs to follow during its execution process.

Mr. Abdul Rehman **email id: abdul@northern.edu.pk** **Whatsapp# 0308-7792217**

BC Instruction cycle: [Fetch Decode [Indirect] Execute]*

• **Fetch and Decode**

T0: AR ← PC (S0S1S2=010, T0=1)
T1: IR ← M [AR], PC ← PC + 1 (S0S1S2=111, T1=1)
T2: D0, . . . , D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)



Namely they are as follow:

1. Fetching an instruction
2. Decoding the instruction
3. Evaluate the effective address
4. Execution

Each instruction has to pass thru these steps. Now we see these steps one by one.

In fetch phase following tasks performed.

T_0: $AR \leftarrow PC$ (initial address is sent to address register from program counter)

T_1: $IR \leftarrow M[AR], PC \leftarrow PC + 1$ (memory address value at AR sent IR, PC incremented)

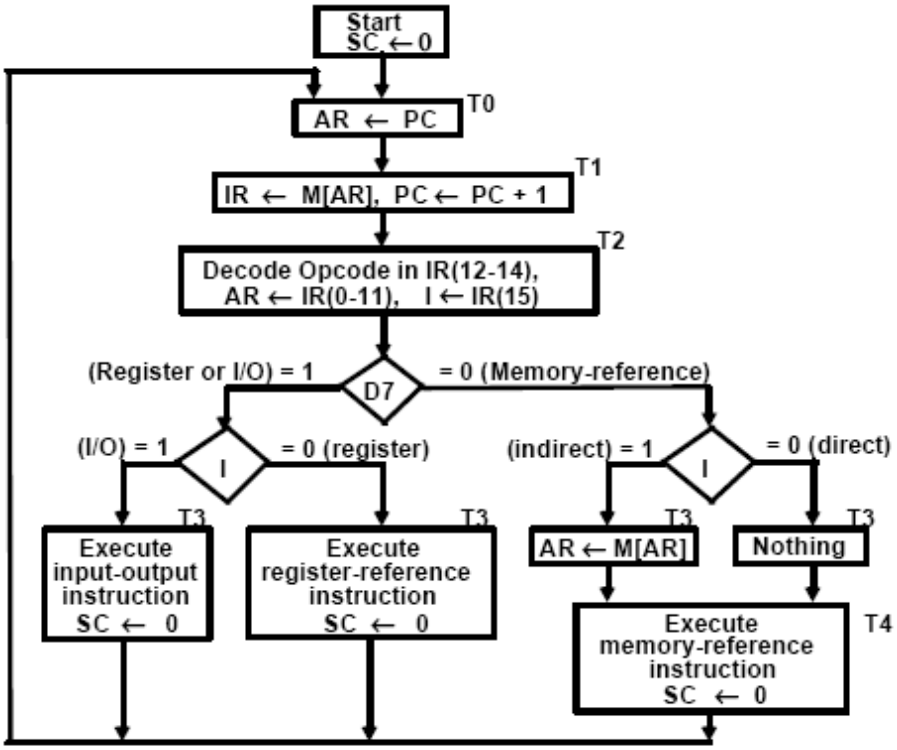
Now to *decode* we go to next time signal T_2,

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

Here the bits 12-14 (opcode) is decoded to the one out of eight levels, also bits of IR from 0-11 (address part) is sent to AR, and 15th bit of IR is sent to I bit. Further timing slots are responsible for execution of the instruction.

TYPE OF INSTRUCTION

This section deals with circuitry which distinguish the type of instruction from the instruction register contents. As we have described earlier that all three type of instructions; memory reference, register reference and I/O have their own distinctions.



D'_7IT_3: $AR \leftarrow M[AR]$
$D'_7I'T_3$: Nothing
$D_7I'T_3$: Execute a register-reference instr.
D_7IT_3: Execute an input-output instr.

From the flowchart given above we can easily understand the type of instruction with respect to time signal associated.

The timing signal that is active after the decoding is T3. During time T3 the control unit determines the type of instruction that was just read from memory.. The three possible instruction types available in the basic computer are specified in Figure above. Decoder output D, is equal to 1 if the operation code is equal to binary figure we determine that if D7 = I, the instruction must be a effective address from memory. The microoperation for the indirect address condition can be symbolized by the register transfer statement

AR←M[AR]

Initially, AR holds the address part of the instruction. This address is used during the memory read operation. The word at the address given by AR is read from memory and placed on the common bus. The LD input of AR is then enabled to receive the indirect address that resided in the 12 least significant bits of the memory word. The three instruction types are subdivided into four separate paths. The selected operation is activated with the clock transition associated with timing signal T3. This can be symbolized as follows:

D7T3: AR←M[AR]

D7T3: Nothing

D7T3: Execute a register-reference instruction

D7T3: Execute an input-output instruction

When a memory-reference instruction with I = 0 is encountered, it is not necessary to do anything since the effective address is already in AR. However, the sequence counter SC must be incremented when D7T3 = 1, so that the execution of the memory-reference instruction can be continued with timing variable T4. A register-reference or input-output instruction can be executed with the clock associated with timing signal T3. After the instruction is executed, SC is cleared to 0 and control returns to the fetch phase with T0 = 1. Note that the sequence counter SC

is either incremented or cleared to 0 with every positive clock transition. We will adopt the convention that if SC is incremented, we will not write the statement SC <-SC + 1, but it will be implied that the control goes to the next timing signal in sequence. When SC is to be cleared, we will include the statement SC <-0. The register transfers needed for the execution of the register-reference instructions are presented in this section.

REGISTER-REFERENCE INSTRUCTIONS

Register-reference instructions are recognized by the control when 07 = 1 and I = 0. These instructions use bits 0 through 11 of the instruction code to specify one of 12 instructions. These 12 bits are available in IR(0-11). They were also transferred to AR during time T2. The control functions and microoperations for the register-reference instructions are listed in the Table below. These instructions are executed with the clock transition associated with timing variable T3. Each control function needs the Boolean relation D7I'T3, which we designate for convenience by the symbol r. The control function is distinguished by one of the bits in IR(0-11). By assigning the symbol B, to bit i of IR, all control functions can be simply denoted by rB;. For example, the instruction CLA has the hexadecimal code 7800 (see Table 5-2), which gives the binary equivalent 0111 1000 0000 0000. The first bit is a zero and is equivalent to I'. The next three bits constitute the operation code and are recognized from decoder output D7. Bit 11 in IR is I and is recognized from B11. The control function that initiates the microoperation for this instruction is D7I'T3B11 = rB11. The execution of a register-reference instruction is completed at time T3. The sequence counter SC is cleared to 0 and the control goes back to fetch the next instruction with timing signal T0.

MEMORY REFERENCE INSTRUCTIONS

These are the instructions which have commonly the code D_7' which shows that opcode is other than 111. Further the '*I'* bit shows the direct or indirect instruction.

Following is the list of such instructions.

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

- The effective address of the instruction is in AR and was placed there during timing signal T_2 when I = 0, or during timing signal T3 when I = 1
- Memory cycle is assumed to be short enough to complete in a CPU cycle
- The execution of MR Instruction starts with T_4

AND to AC

D_0T_4: $DR \leftarrow M[AR]$ Read operand

D_0T_5: $AC \leftarrow AC \wedge DR, SC \leftarrow 0$ AND with AC

ADD to AC

D_1T_4: $DR \leftarrow M[AR]$ Read operand

D_1T_5: $AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$ Add to AC and store carry in E

This is carried out with the timing slots needed.

LDA: Load to AC

D_2T_4: $DR \leftarrow M[AR]$

D_2T_5: $AC \leftarrow DR, SC \leftarrow 0$

STA: Store AC

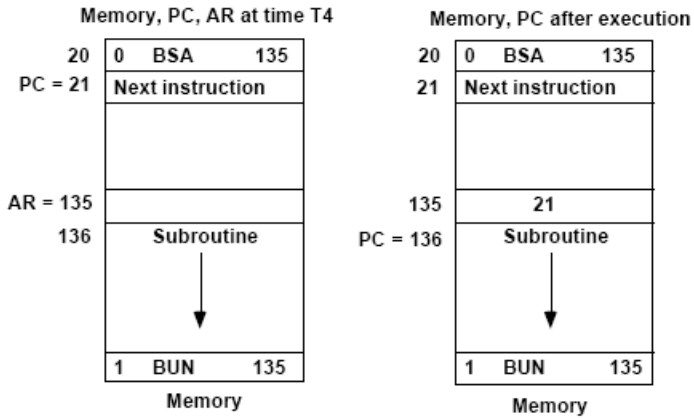
D_3T_4: $M[AR] \leftarrow AC, SC \leftarrow 0$

BUN: Branch Unconditionally

D_4T_4: $PC \leftarrow AR, SC \leftarrow 0$

BSA: Branch and Save Return Address

$M[AR] \leftarrow PC, PC \leftarrow AR + 1$



BSA:

D_5T_4: M[AR] ← PC, AR ← AR + 1
D_5T_5: PC ← AR, SC ← 0

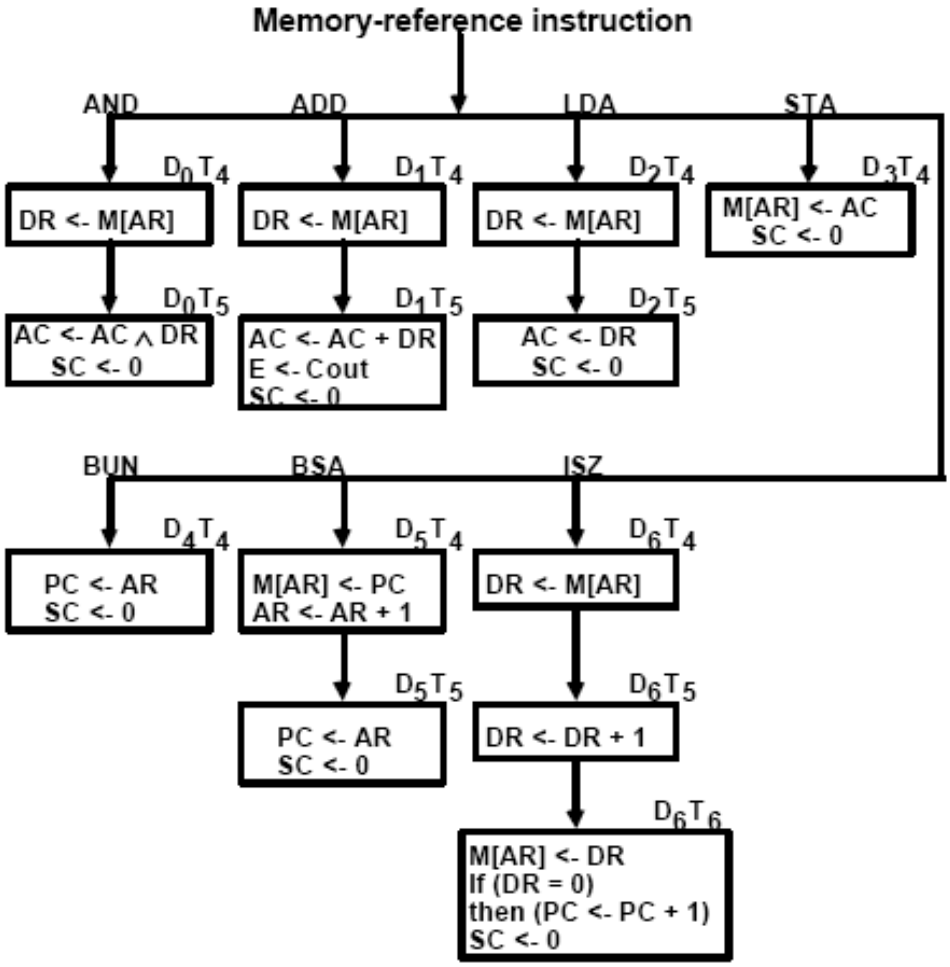
ISZ: Increment and Skip-if-Zero

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

Here branch instructions are of particular interest. If branch is unconditional then in that case we need not to store the return address; while if BSA is the instruction which means branch with saving return address. And after that branch is completed control is returned back to next line branch was occurred. That is demonstrated in figure above.

Following is given a complete flowchart of execution of memory reference instructions, with the type of instruction and timing signal associated.

Mr. Abdul Rehman **email id: abdul@northern.edu.pk** **Whatsapp# 0308-7792217**



$D_7IT_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0–11)$ 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:	$AC \leftarrow$ shr AC, $AC(15) \leftarrow E$, $E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow$ 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