CEG 2136 Computer Architecture I (Fall 2016)

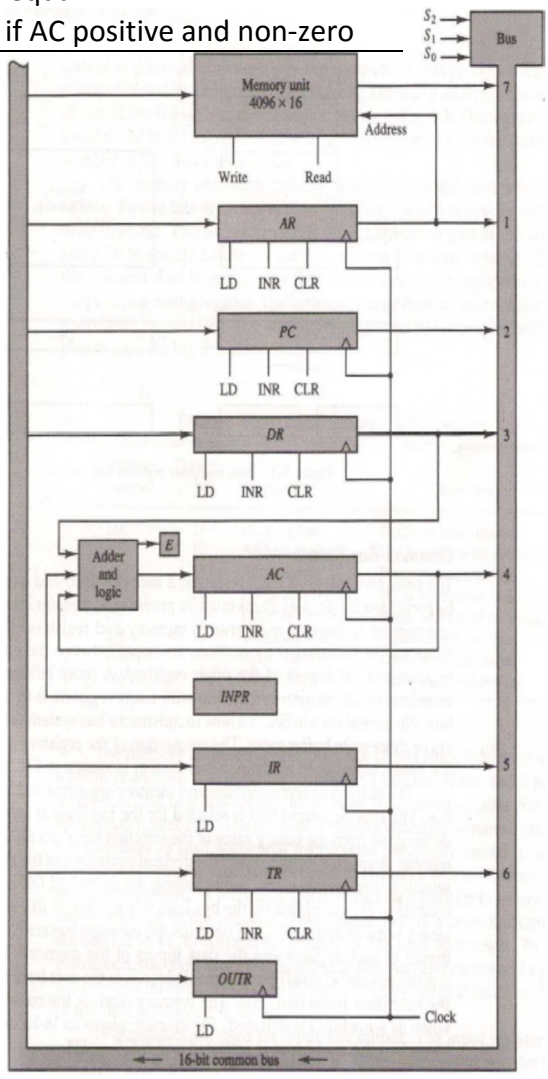
Assignment #3: 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.20, 5.21 from Mano's book

5.13 Assume that the first six memory-reference instructions in the basic computer listed in Table 5-4 are to be changed to the instructions specified in the following table. *EA* is the effective address that resides in *AR* during time T4. Assume that the adder and logic circuit in Fig. 5-4 can perform the exclusive-OR operation $AC \leftarrow AC \oplus DR$. Assume further that the adder and logic circuit cannot perform subtraction directly. The subtraction must be done using the 2' s complement of the subtrahend by complementing and incrementing *AC*. Give the sequence of register transfer statements needed to execute each of the listed instructions starting from timing T4. Note that the value in *AC* should not change, unless the instruction specifies a change in its content. You can use *TR* to store the content of *AC* temporary or you can exchange *DR* and *AC*.

Symbol	Opcode	Symbolic .designation	Description in words
XOR	000	AC←AC ⊕ M[EA]	Exclusive-OR to AC
ADM	001	M[EA]←M[EA] + AC	Add AC to memory
SUB	010	AC←AC-M[EA]	Subtract memory from AC
XCH	011	AC←-M[EA], M[EA] ←AC	Exchange AC and memory
SEQI	100	If (M[EA] = AC) then (PC←PC + 1)	Skip on equal
BPA	101	If (AC > 0) then (PC ←EA)	Branch if AC positive and non-zero

SOLUTION:

XOR	D0 T4	DR ← M[AR]
	D0 T5	AC ← AC ⊕ DR, SC← 0
ADM	D1 T4	DR ← M[AR]
	D1 T5	DR ← AC, AC ← AC + DR
	D1 T5	M[AR] ← AC, AC ← DR, SC← 0
SUB	D2 T4	DR← M[AR]
	D2 T5	DR ← AC, AC ? DR
	D2 T6	AC ← AC'
	D2 T7	AC ← AC + 1
	D2 T8	AC ← AC+DR, SC ← 0
XCH	D3 T4	DR ← M[AR]
	D3 T5	M[AR] ← AC, AC ← DR, SC ← 0
SEQ	D4 T4	DR ← M[AR]
	D4 T5	TR ← AC, AC ← AC ⊕ DR
	D4 T6	If (AC = 0) then (PC ← PC+ 1), AC ← TR, SC ← 0
BPA	D5 T4	If (AC = 0 ∧ AC (15) = 0) then (PC ← AR), SC ← 0



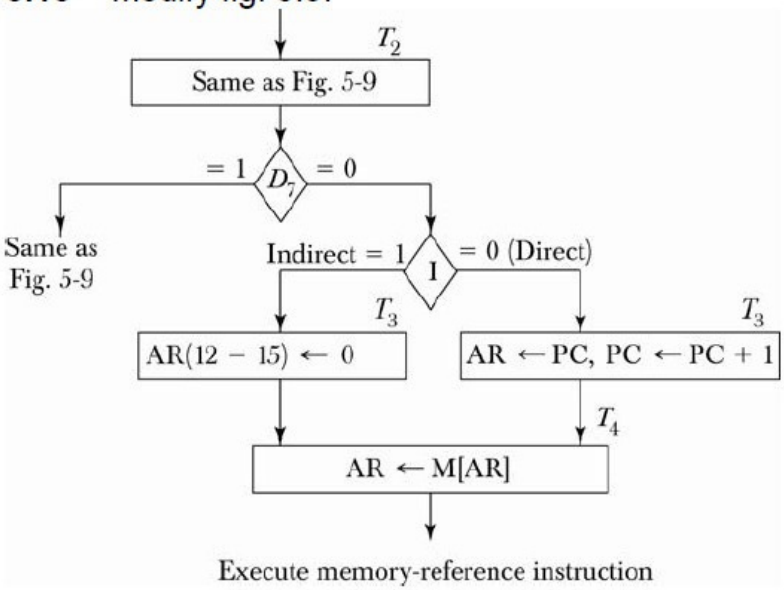
5.14 Make the following changes to the basic computer.

1. Add a register to the bus system CTR (count register) to be selected with $S_2S_1S_0 = 000$.
2. Replace the ISZ instruction with an instruction that loads a number into CTR. LDC Address (*CTR* ←*M [Address]*)
3. Add a register reference instruction ICSZ: Increment *CTR* and skip next instruction if zero. Discuss the advantage of this change.

Converts the ISZ instruction from a memory-reference instruction to a register- reference instruction. The new instruction ICSZ can be executed at time T3, T4 instead of time T6.

5.15 The memory unit of the basic computer shown in Fig, 5-3 is to be changed to a *65,536* X 16 memory, requiring an address of 16 bits. The instruction format of a memory-reference instruction shown in Fig. 5-S(a) remains the same for $I = 1$ (indirect address) with the address part of the instruction residing in positions 0 through 11, But when $I = 0$ (direct address), the address of the instruction is given by the 16 bits in the next word following the instruction. Modify the microoperations during time T2, *T3,* (and *T4* if necessary) to conform with this configuration.

5.15 Modify fig. 5.9.



5.16 A computer uses a memory of 65,536 words with eight bits in each word. It has the following registers: *PC, AR, TR* (16 bits each), and *AC, DR, IR* (eight bits each). A memory-reference instruction consists of three words: an 8-bit operation-code (one word) and a 16-bit address (in the next two words). All operands are eight bits. There is no indirect bit.

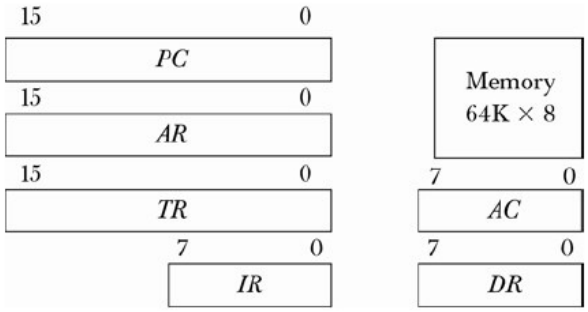
a. Draw a block diagram of the computer showing the memory and registers as in Fig. S-3. (Do not use a common bus).

b. Draw a diagram showing the placement in memory of a typical three word instruction and the corresponding 8-bit operand.

c. List the sequence of micro-operations for fetching a memory reference instruction and then placing the operand in *DR.* Start from timing signal *To.*

5.16

(a)



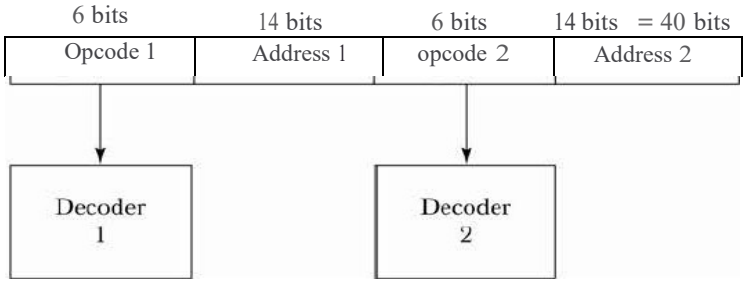
(b)

c)

T0: IR←M(PC), PC ←PC+ 1
T1: AR(0-7) ← M[PC], PC ←PC+ 1
T2: AR(8-15) ←M[PC], PC ← PC +1
T3: DR ← M [AR]

5.17 A digital computer has a memory unit with a capacity of 16,384 words, 40 bits per word. The instruction code format consists of six bits for the operation part and 14 bits for the address part (no indirect mode bit). Two instructions are packed in one memory word, and a 40-bit instruction register IR is available in the control unit. Formulate a procedure for fetching and executing instructions for this computer.

IR:



1. Read 40-bit double instruction from memory to IR and then increment PC.
2. Decode opcode 1.
3. Execute instruction 1 using address 1.
4. Decode opcode 2.
5. Execute instruction 2 using address 2.
6. Go back to step 1.

5.18 An output program resides in memory starting from address 2300. It is executed after the computer recognizes an interrupt when *FGO* becomes a 1 (while *IEN*= 1).

a. What instruction must be placed at address 1?

b. What must be the last two instructions of the output program?

(a) BUN 2300

(b) ION

BUN 0 I (Branch indirect with address 0)

5.20 The operations to be performed with a flip-flop *F* (not used in the basic computer) are specified by the following register transfer statements:

xT3: $F \leftarrow 1$ Set *F* to 1

yT1: $F \leftarrow 0$ Set *F* to 0

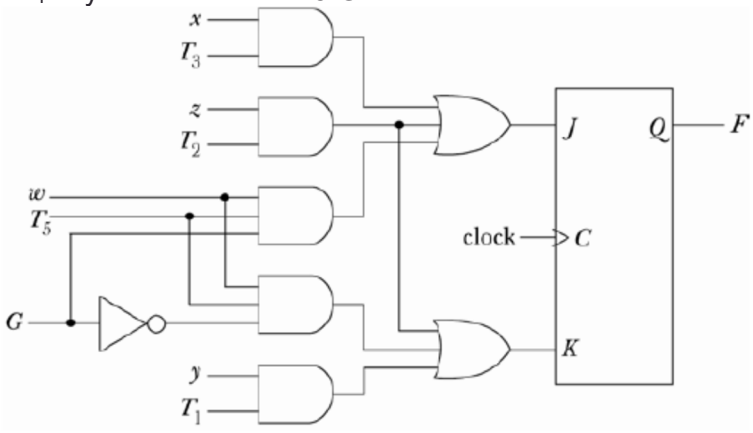
zT2: $F \leftarrow F'$ Complement *F*

wT3: $F \leftarrow G$ Transfer value of *G* to *F*

Otherwise, the content of *F* must not change. Draw the logic diagram showing the connections of the gates that form the control functions and the inputs of flip-flop *F*. Use a *JK* flip-flop and minimize the number of gates.

$J_F = x\,T3 + z\,T2 + w\,T5\,G$

$K_F = y\,T1 + z\,T2 + w\,T5\,G'$



5.21 Derive the control gates associated with the program counter *PC* in the basic computer.

From Table 5.6:
(Z_{DR} = 1 if DR = 0; Z_{AC} = 1, if AC = 0)

INR (PC) = R'T_1 + RT_2+ D_6 T_6 Z_{DR} + rB_4 (AC15)' + rB_3 (AC15) + rB_2 Z_{AC} + rB_1E' + p B_9 (FGI) +p B_8 (FGO)

LD (PC) = D_4 T_4 + D_5 T_5

CLR(PC) = RT_1

The logic diagram is similar to the one in Fig. 5.16.

Fetch	$R'T_0$:	*AR <- PC*
	$R'T_1$:	*IR <- M[AR], **PC <- PC + 1***
Decode	$R'T_2$:	*D_0, ... , D_7 <- Decode IR(12-14),*
		AR <- IR(0-11), I <- IR(15)
Indirect	D'_7IT_3:	*AR <- M[AR]*
Interrupt		
IEN $\overline{T}_0\overline{T}_1\overline{T}_2$*(FGI+FGO)*:		*R <- 1*
	RT_0:	*AR <- 0, TR <- PC*
	RT_1:	*M[AR] <- TR, **PC <- 0***
	RT_2:	***PC <- PC+1**, IEN <- 0, R <- 0, SC <- 0*
	Execute	**Memory-reference Instructions (MRI)**
AND	D_0T_4:	*DR <- M[AR]*
	D_0T_5:	*AC <- AC ^ DR, SC <- 0*
ADD	D_1T_4:	*DR <- M[AR]*
	D_1T_5:	*AC <- AC + DR, E <- C_{out}, SC <- 0*
LDA	D_2T_4:	*DR <- M[AR]*
	D_2T_5:	*AC <- DR, SC <- 0*
STA	D_3T_4:	*M[AR] <- AC, SC <- 0*
BUN	D_4T_4:	***PC <- AR**, SC <- 0*
BSA	D_5T_4:	*M[AR] <- PC, AR <- AR + 1*
	D_5T_5:	***PC <- AR**, SC <- 0*
ISZ	D_6T_4:	*DR <- M[AR]*
	D_6T_5:	*DR <- DR + 1*
	D_6T_6:	*M[AR] <- DR,*
	D_6T_6DR':	*if (**DR = 0**) then (**PC <- PC + 1**), SC <- 0*
	Execute	**Register-reference Instructions (RRI)**
	$D_7I'T_3$ = **r**	(common to all Register-ref. instructions)
		IR(i) = B_i (i = 0, 1, 2, ... , 11)
	r:	SC <- 0
CLA	rB_{11}:	*AC <- 0*
CLE	rB_{10}:	*E <- 0*
CMA	rB_9:	*AC <- AC'*
CME	rB_8:	*E <- E'*
CIR	rB_7:	*AC <- shr AC, AC(15) <- E, E <- AC(0)*
CIL	rB_6:	*AC <- shlAC, AC(0) <- E, E <- AC(15)*
INC	rB_5:	*AC <- AC + 1*
SPA	rB_4AC'(15):	*If (AC(15) = 0) then (**PC <- PC + 1**)*
SNA	rB_3AC(15):	*If (AC(15) = 1) then (**PC <- PC + 1**)*
SZA	rB_2AC':	*If (AC = 0) then (**PC <- PC + 1**)*
SZE	rB_1E':	*If (E = 0) then (**PC <- PC + 1**)*
HLT	rB_0:	*S <- 0*
	Execute	**Input-output (IOI)**
	D_7IT_3 = *p*	(common to all input-output instructions)
	IR(i) = B_i	(i = 6, 7, 8, 9, 10, 11)
	p:	*SC <- 0*
INP	pB_{11}:	*AC(0-7) <- INPR, FGI <- 0*
OUT	pB_{10}:	*OUTR <- AC(0-7), FGO <- 0*
SKI	pB_9FGI:	*If (FGI = 1) then (**PC <- PC + 1**)*
SKO	pB_8FGO:	*If (FGO = 1) then (**PC <- PC + 1**)*
ION	pB_7:	*IEN <- 1*
IOF	pB_6:	*IEN <- 0*