University of Oulu
ee.oulu.fi
Electrical and Information Engineering

EXERCISE #3

PROBLEM 3.1(5-1)

A computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers, and an address part.

a) How many bits are there in the operation code, the register code part, and the address part?

b) Draw the instruction word format and indicate the number of bits in each part.

c) How many bits are there in the data and address inputs of the memory.

	Memory
0	32 bits
1	
...	
256K	32 bits

Instruction word format

I	OP-code	Register	Address

Solution:

a)

Indirect	1 bit
Address	2^8 (256kB) * 2^{10} (1024 bytes/kB) = 2^{18} ==> 18 bits
Reg	64 registers = 2^6 => 6 bits
OP-code	32 - 1 - 18 - 6 bits = 7 bits

b)

I 31	OP 24-30	Reg 18-23	Address 0-17

c)

Address inputs: 18 (256K words)

data inputs: 32 (32 bits / word / memory reference)

PROBLEM 3.2(5-2)

What is the difference between a direct and an indirect address instruction? How many references to memory are needed for each type of instruction to bring an operand into a processsor register?

Solution:

Direct:

One memory reference: Get the operand from the given address

Indirect:

Two memory references: Get the address of the operand from the given address and use this new address to get the value of the operand

PROBLEM 3.3(5-3)

The following control inputs are active in the bus system shown in Fig. 5-4. For each case, specify the register transfer that will be executed during the next clock transition.

See figure 5-21 for a functionality description of the Adder.

	S2	S1	S0	value S	LD(x)	Memory	Adder
a)	1	1	1	=7	IR	READ	-
b)	1	1	0	=6	PC	-	-
c)	1	0	0	=4	DR	WRITE	-
d)	0	0	0	=0	AC	-	ADD

Solution:

See Fig. 5-4

a)

```
BUS(7) => Connect memory to the BUS
READ   => Read from memory
LD(IR) => Load from bus to IR
Result :  IR <- M[AR]
```

b)

```
BUS(6) => Connect register TR to the BUS
LD(PC) => Load from bus to PC
Result :  PC <- TR
```

c)

```
BUS(4) => Connect register AC to the BUS
WRITE  => Write to the memory
LD(DR) => Load from bus to DR
Result :  M[AR] <- AC, DR <- AC
```

d)

```
BUS(0) => No effect
```

ADD => Add DR to AC

Result : AC <- AC + DR

PROBLEM 3.4(5-4)

The following register transfers are to be executed in the system of Fig. 5-4. For each transfer, specify: (1) the binary value that must be applied to bus select inputs S_2, S_1 and S_0; (2) the register whose LD control input must be active (if any); (3) a memory read or write operation (if needed); and (4) the operation in the adder and logic circuit (if any).

a) AR <- PC

b) IR <- M[AR]

c) M[AR] <- TR

d) AC <- DR, DR <- AC

Solution:

See fig. 5-4

a) AR <- PC

S2	S1	S0	value S	LD(x)	Memory	Adder
0	1	0	=2	AR	-	-

S = 2: Connect PC to the bus

b) IR <- M[AR]

S2	S1	S0	value S	LD(x)	Memory	Adder
1	1	1	=7	IR	READ	-

S = 7: Connect memory to the bus

c) M[AR] <- TR

S2	S1	S0	value S	LD(x)	Memory	Adder
1	1	0	=6	-	WRITE	-

S = 6: Connect TR to the bus

d) AC <- DR, DR <- AC

S2	S1	S0	value S	LD(x)	Memory	Adder
1	0	0	=4	AC, DR	-	DR

S = 4: Connect AC to the bus

PROBLEM 3.5(5-5)

Why more than one clock period is needed to execute following microoperations?

a) IR <- M[PC]

b) AC <- AC + TR

c) DR <- DR + AC (AC not changed!)

Solution:

See fig. 5-4

```
a)
AR <- PC ; Address from PC to AR
IR <- M[AR] ; Move word from the memory

b)
DR <- TR ; Second operand to the DR
AC <- AC + DR ; Create sum

c)
E.g.
TR <- AC ; Save AC
AC <- AC + DR ; Create sum
DR <- TR ; Restore AC to DR
AC <- DR, DR <- AC ; Move old AC to AC, result (DR+AC) to DR
```

PROBLEM 3.6(5-6)

Consider the instruction formats of the basic computer shown in Fig. 5-5 and the list of instructions given in Table 5-2. For each of the following 16-bit instructions, give the equivalent four-digit hexadecimal code and explain in your own words what it is that the instruction is going to perform.

a) 0001 0000 0010 0100

b) 1011 0001 0010 0100

c) 0111 0000 0010 0000

Solution:

a)

Instruction	0001	0000	0010	0100
Hex.	1	0	2	4

=> direct ADD:

AC <- M[AR] + AC ; Address = 24

b)

Instruction	1011	0001	0010	0100
Hex.	B	1	2	4

=> indirect STA:

AR<-M[AR]

M[AR] <- AC ; Address is found at address 124

c)

Instruction	0111	0000	0010	0000
Hex.	7	0	2	0

=> INC:

AC <- AC + 1

PROBLEM 3.7(5-9)

The content of AC in the basic computer is hexadecimal A937 and the initial value of E is 1. Determine the contents of AC, E, PC, AR and IR in hexadecimal after the execution of the CLA instruction. The initial value of PC is hexadecimal 021.

Initial conditions:

AC = A937h
E = 1

PC = 021h

See figure 5-6 for hints

Solution:

Execute CLA (Instruction code 7*800*h)

```
Fetch:
AR <- PC                          ; => (AR) = (PC) = 021h
IR <- M[AR] , PC <- PC + 1   ; => (IR)=7800h, (PC) = 022h

Decode:
D0,...,D7 <- decode IR(12-14)
AR <- IR(0-11), I <- IR(15) ; =>(AR) = 800h (12 address bits)

Execute:
AC <- 0                           ; (AC) = 0
Summary:
 (AC) = 0h, (AR) = 800h, (PC) = 022h, (IR) = 7800h, E= 1
```

PROBLEM 3.8(5-7)

A basic computer is starting to perform instruction **ADD 100 I**. Given preconditions are (values are hex decimals):

- PC = 190
- AC = 3
- M[100] = 200
- M[200] = fffe

a) Describe what happens during the instruction cycle. Include all phases from fetch to execute.

b) If an I/O device requests for an interrupt during the instruction cycle, what happens? Describe the events starting from the fetch phase of the current instruction until the machine is ready to branch to the interrupt subroutine of the I/O device.

See figure 5-15.

Solution:

a)

Before fetch, we branch to the instruction cycle (R = 0).

```
Fetch:
R'T0:   AR <- PC                        ; AR = PC = 190
R'T1:   IR <- M[AR] , PC <- PC + 1      ; IR = 9100, PC = 191

Decode:
R'T2:   D0,...,D7 <- decode IR(12-14)   ; D1 = 1
        AR <- IR(0-11), I <- IR(15)     ; I = 1, AR = 100
```

Determine whether instruction is memory reference, register or I/O. Because D1 was set to 1, all other decoder outputs Dn are set to 0 (see figure 5-6). Therefore the instruction is a memory-reference instruction.

```
Indirect:
D7'IT3: AR <- M[AR]                     ; AR = 200

Execute:
D1T4:   DR <- M[AR]                     ; DR = fffe
D1T5:   AC <- AC + DR,                  ; AC = 1
        E <- C(out), SC <- 0            ; E = 1
```

Remember that SC (Sequence Counter) is incremented on every clock pulse Tn.

b)

Because we are already in the instruction cycle, the current instruction is performed as in a). After the execution of the instruction is completed, the computer branches to interrupt cycle (R = 1). R is the interrupt request signal.

First in the interrupt cycle, the return address of the current program is saved to memory location 0.

```
RT0: AR <- 0, TR <- PC
RT1: M[AR] <- TR, PC <- 0           ; M[0] = 191
```

The interrupts are disabled and the program flow continues at memory location 1 which contains the branch instruction to the interrupt subroutine.

```
RT2: PC <- PC + 1 , IEN <- 0,
     R <- 0, SC <- 0
```

At the beginning of the next instruction cycle (R = 0), fetching the branch instruction to the first instruction of the interrupt subroutine may begin.

PROBLEM 3.9(5-12)

The content of PC in the basic computer is 3AF (all numbers are in hexadecimal). The content of AC is 7EC3. The content of memory at address 3AF is 932E. The content of memory at address 32E is 09AC. The content of memory at address 9AC is 8B9F.

a) What is the instruction that will be fetched and executed next?

b) Show the binary operation that will be performed in the AC when the instruction is executed

c) Give the contents of registers PC, AR, DR, AC and IR in hexadecimal and the values of E, I and the sequence counter SC in binary at the end of the instruction cycle.

Solution:

a) (PC) = 3AF => next instruction that will be fetched is 9*32E* that is indirect ADD instruction (from table 5-2). The address of the operand is at address *32E* (instruction bits 0 - 11).

b) Address of the operand is 09AC at address 32E => operand is at address 09AC => operand is 8B9F

ADD command:

```
DR <- M[AR]
AC <- AC + DR ; operand is in DR
E <- Cout , SC <- 0 ; E = 1 if carry out

=> (AC) = (AC) + 8B9F = 7EC3 + 8B9F
=>

  7EC3 = 0111 1110 1100 0011
+ 8B9F = 1000 1011 1001 1111
--------------------------------
       1 0000 1010 0110 0010 = 0A62 (carry discarded)
```

c)

```
(PC) = 3B0  ; 3AF + 1
(AR) = 9AC  ; Address of the last memory reference
(DR) = 8B9F ; 2'nd operand
(AC) = 0A62 ; The result of addition
(IR) = 932E ; Last instruction fetched = ADD
 E = 1      ; we had carry out
(SC) = 0    ; Cleared at the end of ADD command
 I = 1      ; indirect ADD
```

PROBLEM 3.10(5-22)

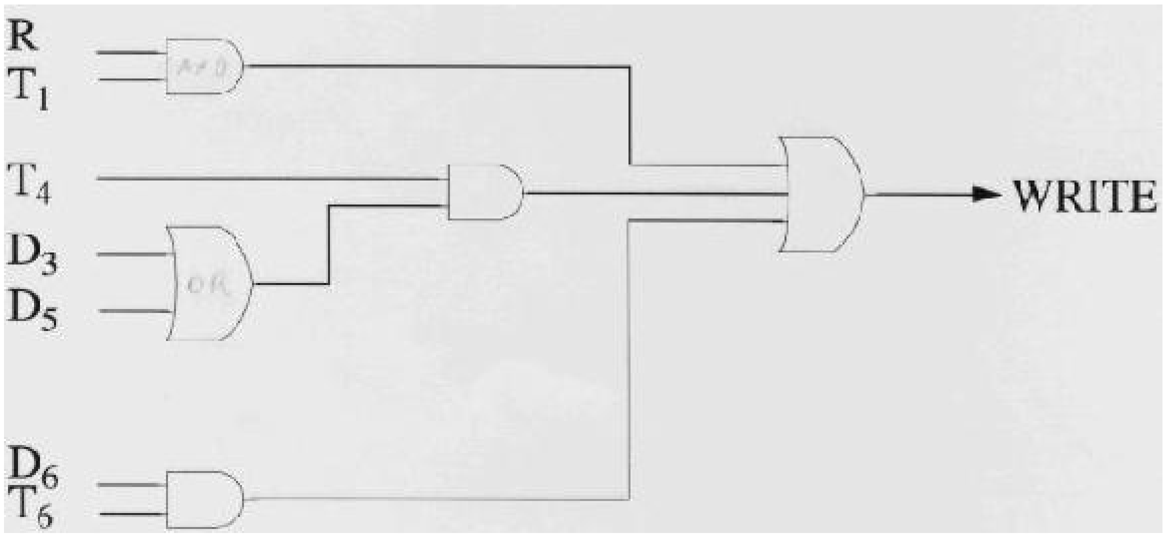
Derive the control gates for the write input of the memory in the basic computer.

See Table 5-6

Solution:

The logic gates associated with the write input of memory is derived by scanning Table 5-6 to find the statements that specify a write operation. The write operation is recognized from the symbol M[AR] <-.

```
WRITE = RT1 + D3T4 + D5T4 + D6T6
= RT1 + T4( D3 + D5 ) + D6T6
```



PROBLEM 3.11 (5-19)

The register transfer statements for a register R and the memory in a computer are as follows (the X's are control functions that occur at random):

X'_3X1: R<-M[AR] Read memory word into R

X'_1X2: R<-AC Transfer AC to R

X'_1X3: M[AR]<-R Write R to memory

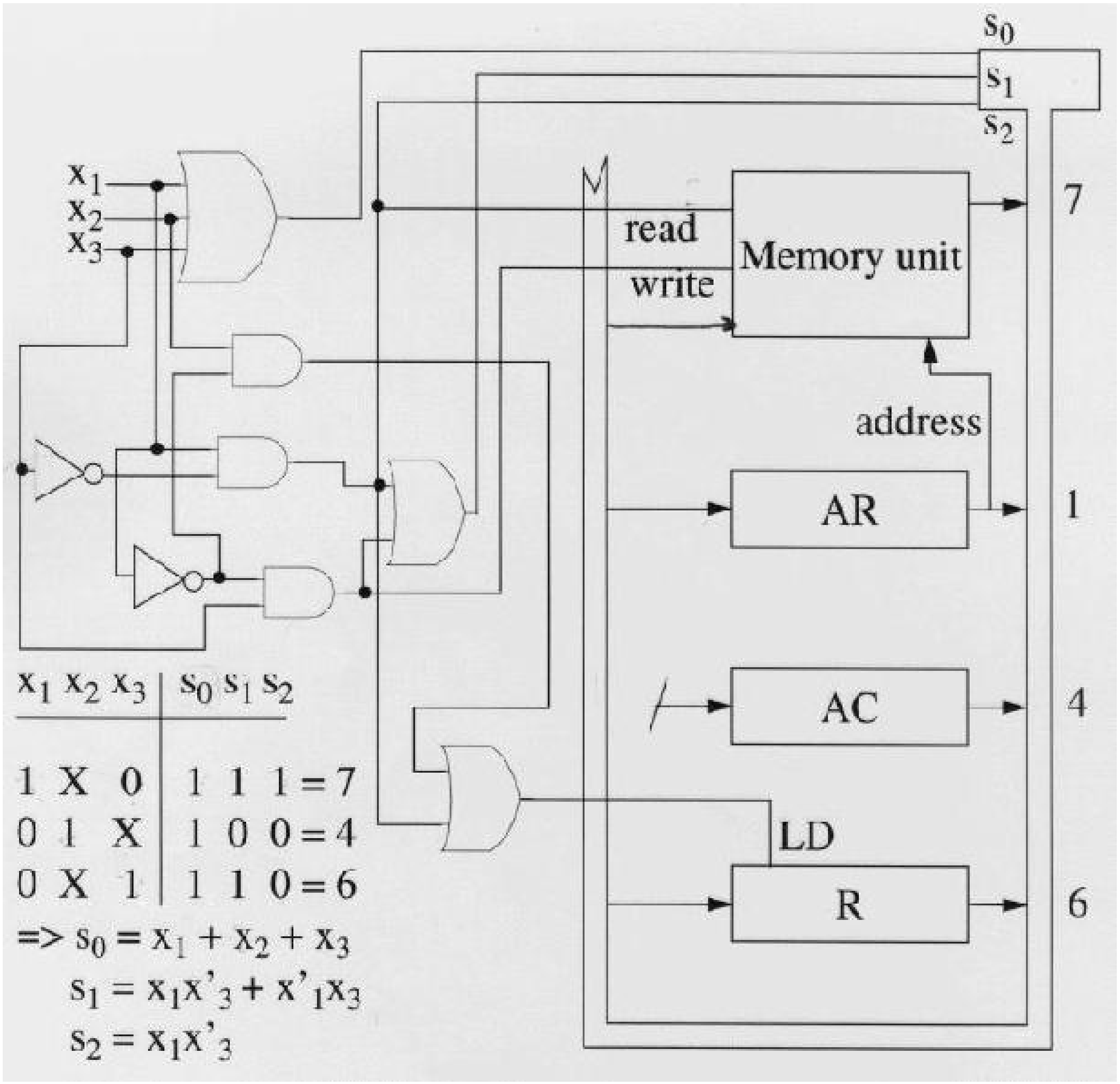
The memory has data inputs, data outputs, address inputs and control inputs to read and write as in Fig. 2-12. Draw the hardware implementation of R and the memory in block diagram form. Show how the control functions X_1 through X_3 select the load control input of R, the select inputs of multiplexers that you include in the diagram, and the read and write inputs of the memory.

See fig. 5-4

Solution:

Following statements change the contents of R (LD = 1):

```
x'3x1: R <- M[AR]
x'1x2: R <- AC
=> LD(R) = x1x'3 + x'1x2
```



x_1	x_2	x_3	s_0	s_1	s_2
1	X	0	1	1	1 = 7
0	1	X	1	0	0 = 4
0	X	1	1	1	0 = 6

$$\Rightarrow s_0 = x_1 + x_2 + x_3$$
$$s_1 = x_1x'_3 + x'_1x_3$$
$$s_2 = x_1x'_3$$

Appendices

Figure 2-12

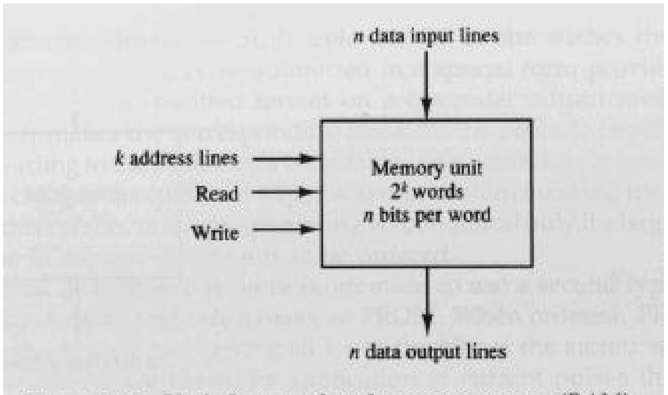


Figure 2-12 Block diagram of random access memory (RAM).

Figure 5-4

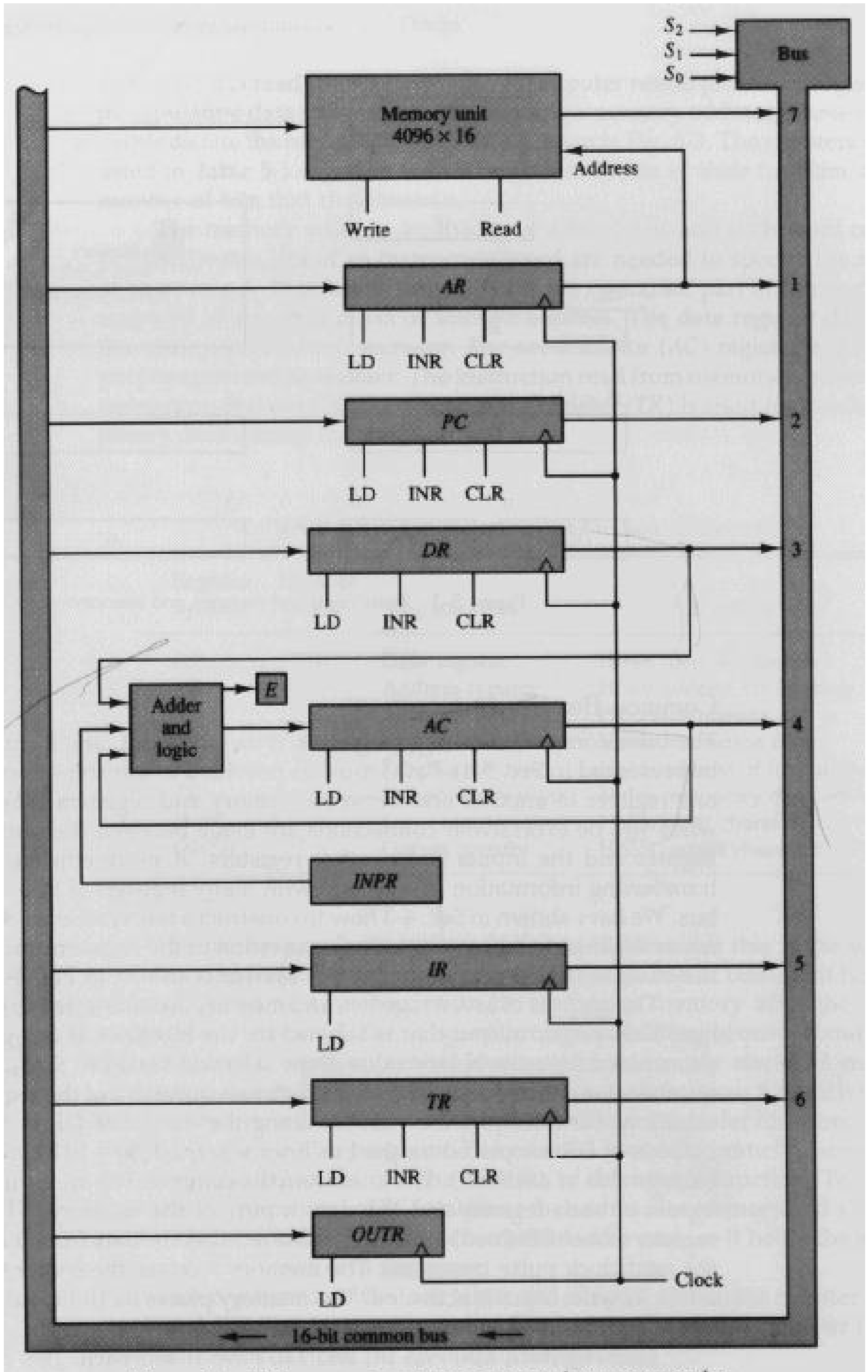


Figure 5-4 Basic computer registers connected to a common bus.

Figure 5-5

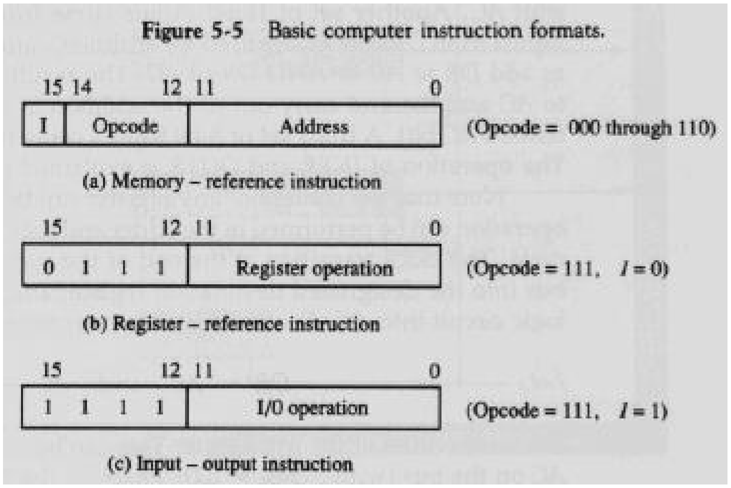


Figure 5-5 Basic computer instruction formats.
15 14 12 11 0
I
Opcode
Address
(Opcode = 000 through 110)
(a) Memory – reference instruction
15 12 11 0
0 1 1 1
Register operation
(Opcode = 111, I = 0)
(b) Register – reference instruction
15 12 11 0
1 1 1 1
I/0 operation
(Opcode = 111, I = 1)
(c) Input – output instruction

Figure 5-6

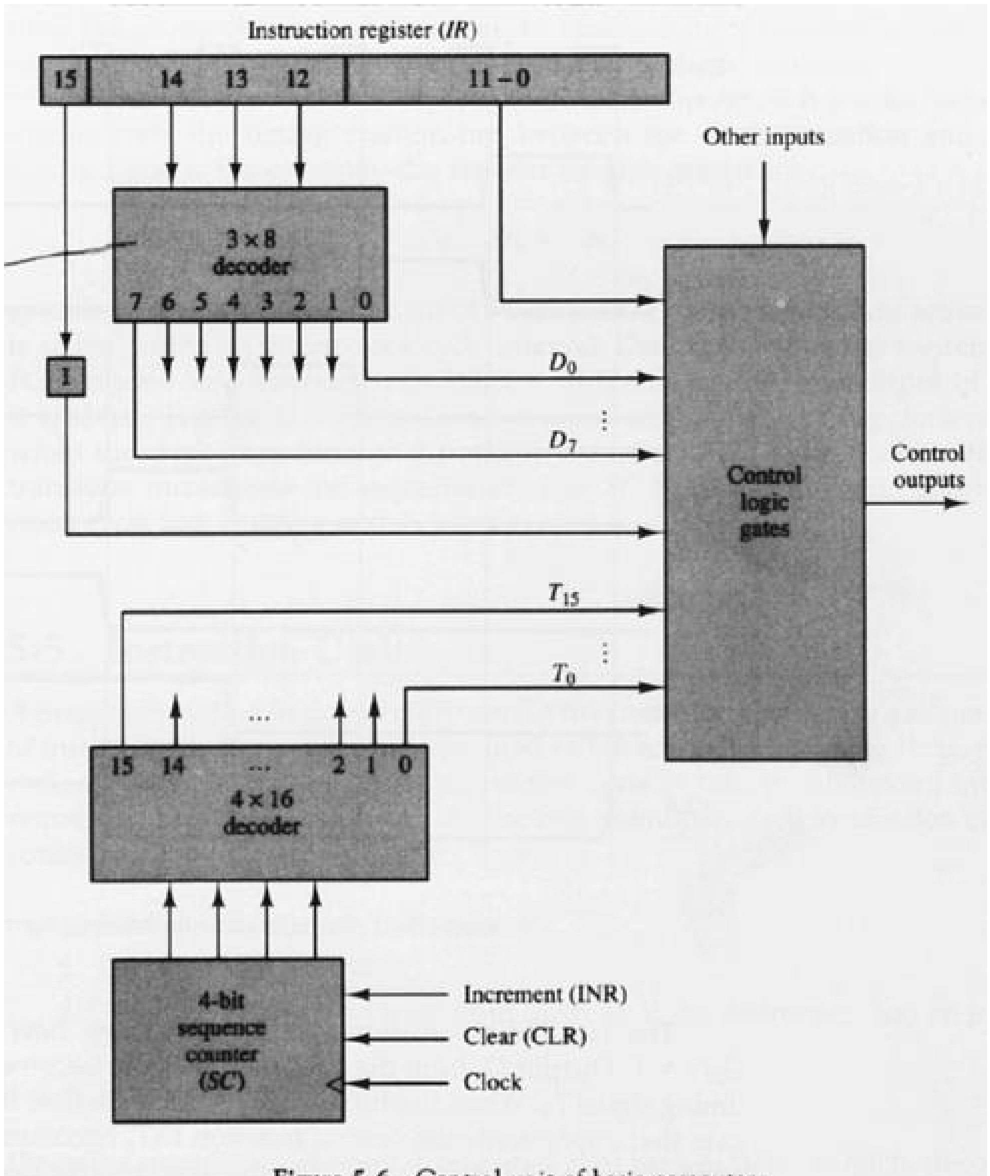


Figure 5-6 Control unit of basic computer.

Figure 5-15

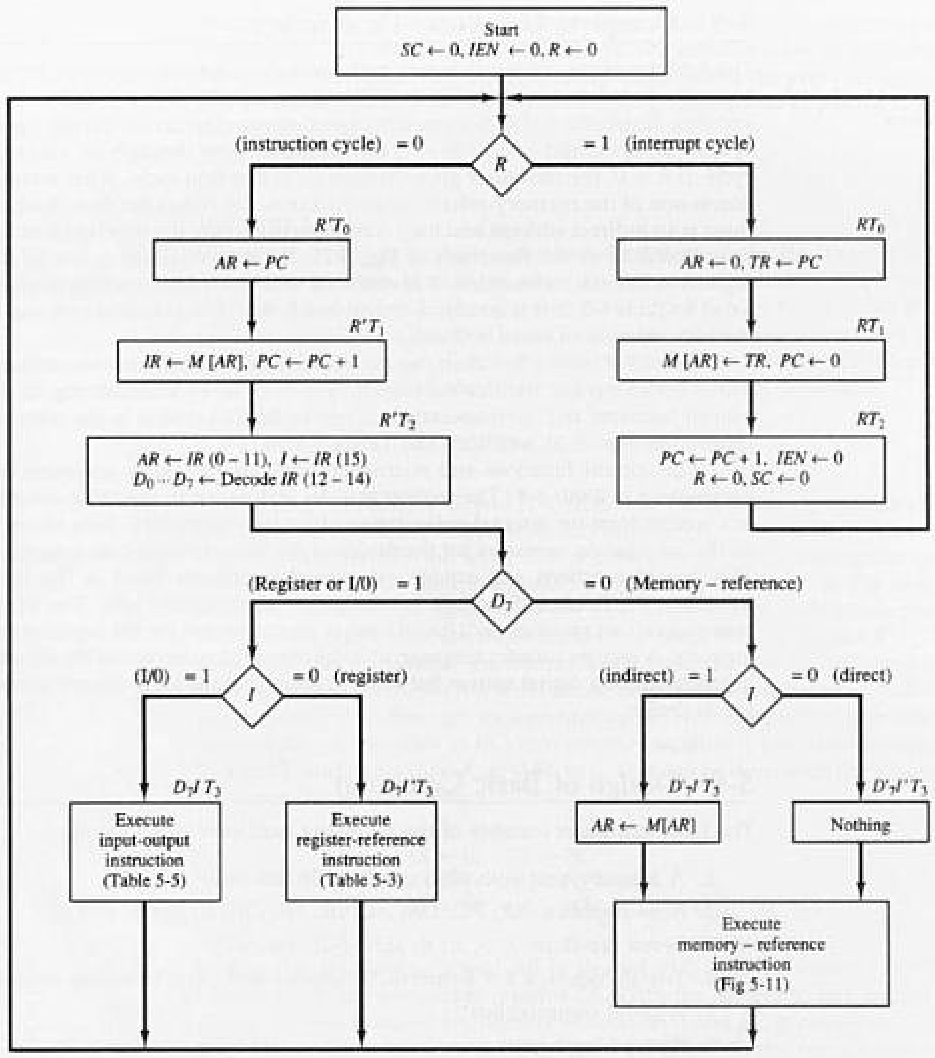


Figure 5-15 Flowchart for computer operation.

Figure 5-21

Figure 5-21 One stage of adder and logic circuit.

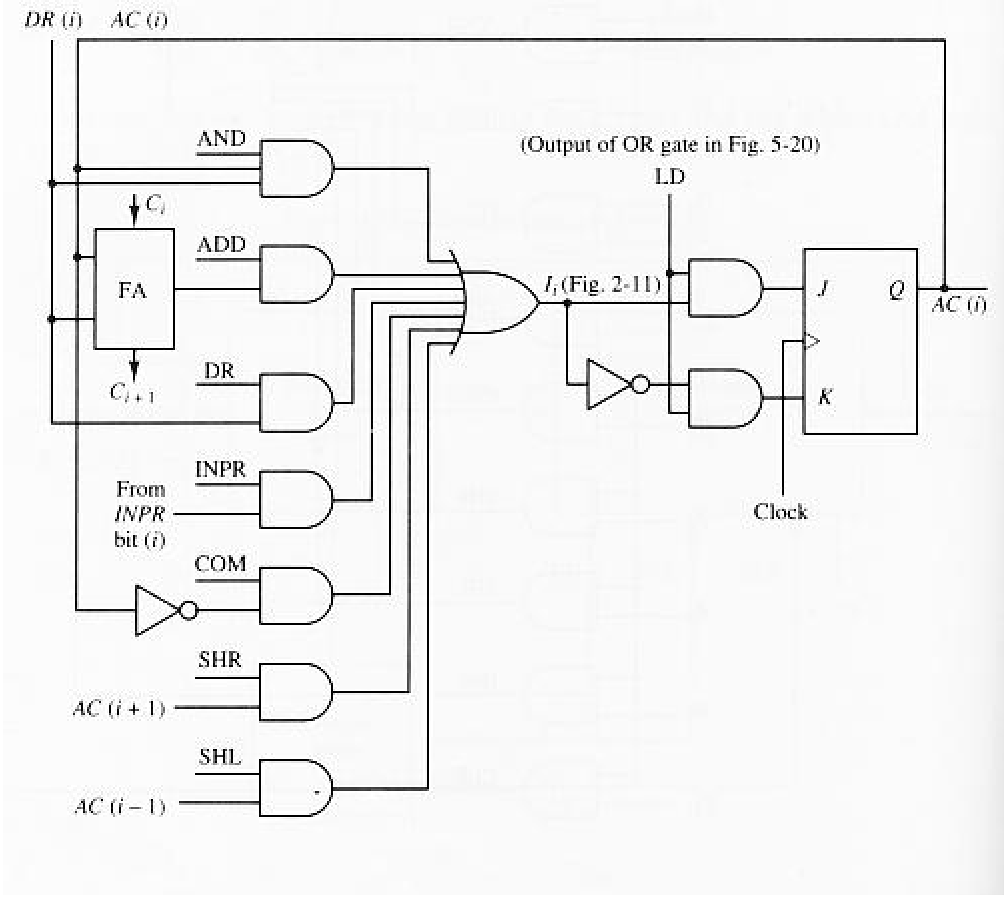


Table 5-2

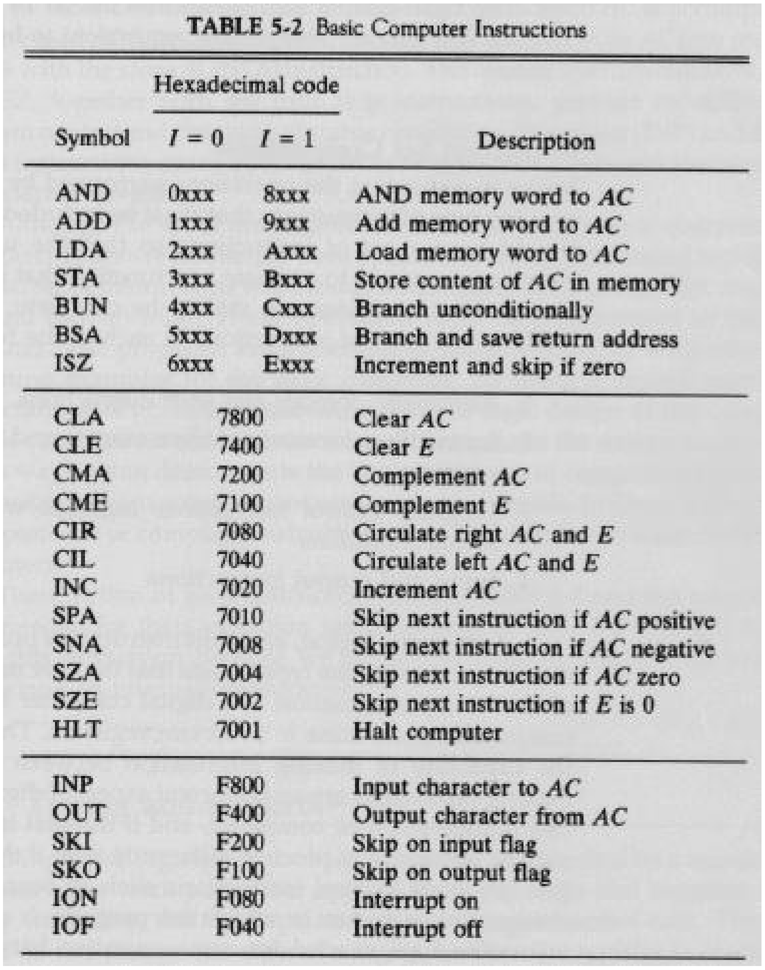


TABLE 5-2 Basic Computer Instructions

	Hexadecimal code		
Symbol	$I = 0$	$I = 1$	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		Complement *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		Interrupt on
IOF	F040		Interrupt off

Table 5-4

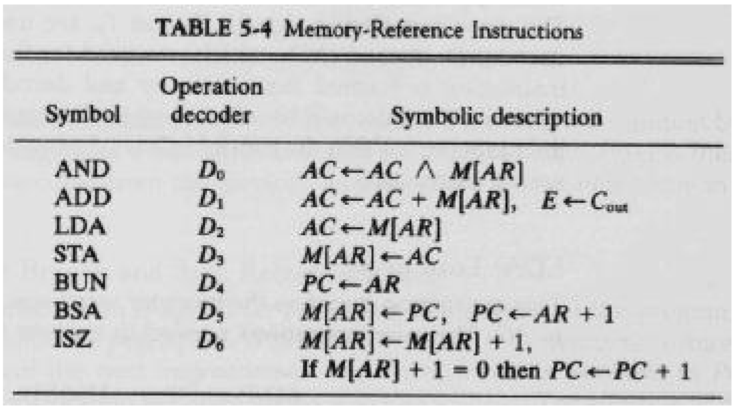


TABLE 5-4 Memory-Reference Instructions

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 $PC \leftarrow PC + 1$

Table 5-6

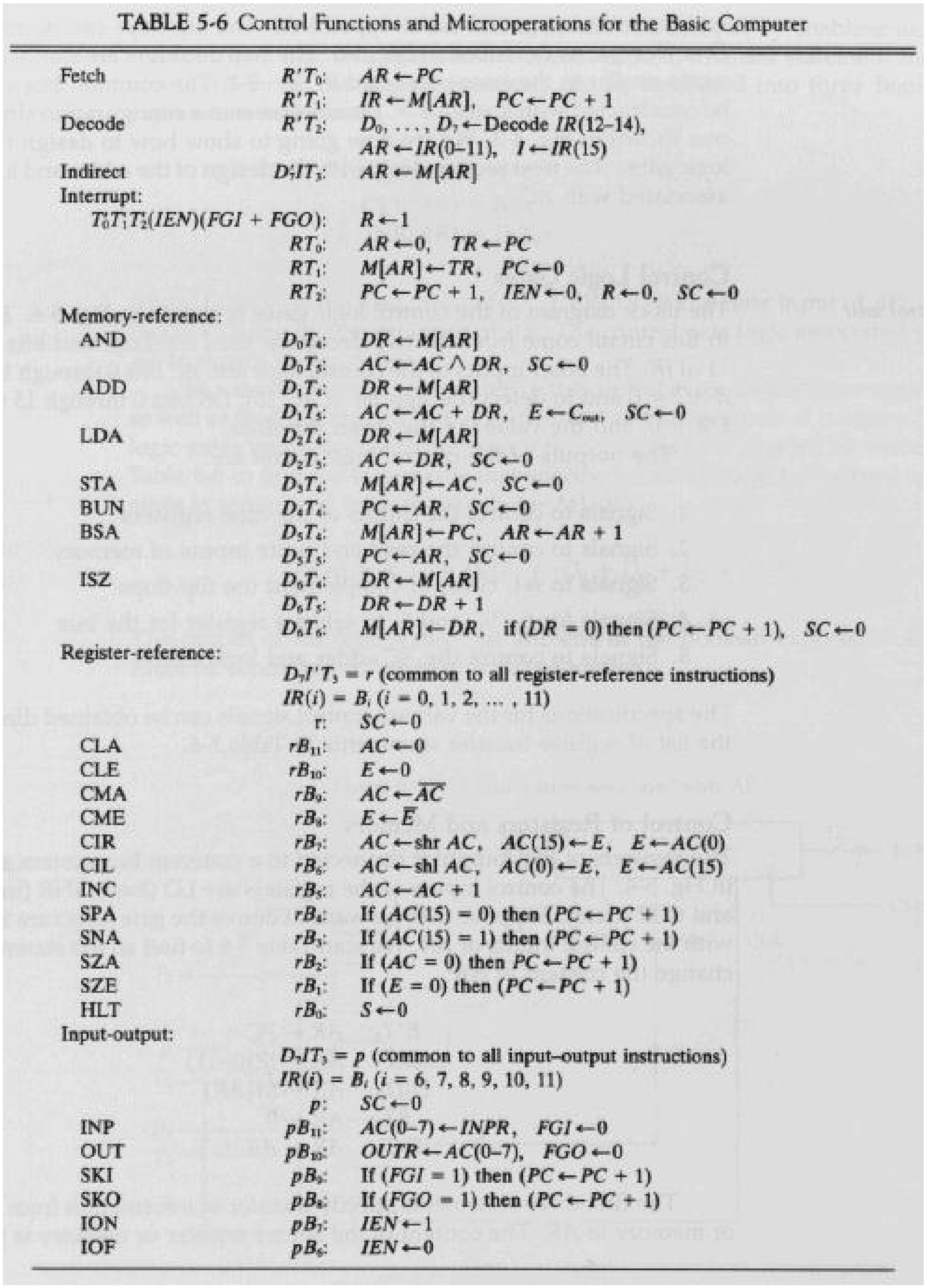


TABLE 5-6 Control Functions and Microoperations for the Basic Computer

Fetch	$R'T_0$:	$AR \leftarrow PC$
	$R'T_1$:	$IR \leftarrow M[AR], \quad PC \leftarrow PC + 1$
Decode	$R'T_2$:	$D_0, \ldots, D_7 \leftarrow$ Decode $IR(12\text{–}14)$,
		$AR \leftarrow IR(0\text{–}11), \quad I \leftarrow IR(15)$
Indirect	$D_7'IT_3$:	$AR \leftarrow M[AR]$
Interrupt:		
$T_0'T_1'T_2'(IEN)(FGI + FGO)$:		$R \leftarrow 1$
	RT_0:	$AR \leftarrow 0, \quad TR \leftarrow PC$
	RT_1:	$M[AR] \leftarrow TR, \quad PC \leftarrow 0$
	RT_2:	$PC \leftarrow PC + 1, \quad IEN \leftarrow 0, \quad R \leftarrow 0, \quad SC \leftarrow 0$
Memory-reference:		
AND	D_0T_4:	$DR \leftarrow M[AR]$
	D_0T_5:	$AC \leftarrow AC \wedge DR, \quad SC \leftarrow 0$
ADD	D_1T_4:	$DR \leftarrow M[AR]$
	D_1T_5:	$AC \leftarrow AC + DR, \quad E \leftarrow C_{out}, \quad SC \leftarrow 0$
LDA	D_2T_4:	$DR \leftarrow M[AR]$
	D_2T_5:	$AC \leftarrow DR, \quad SC \leftarrow 0$
STA	D_3T_4:	$M[AR] \leftarrow AC, \quad SC \leftarrow 0$
BUN	D_4T_4:	$PC \leftarrow AR, \quad SC \leftarrow 0$
BSA	D_5T_4:	$M[AR] \leftarrow PC, \quad AR \leftarrow AR + 1$
	D_5T_5:	$PC \leftarrow AR, \quad SC \leftarrow 0$
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 $(PC \leftarrow PC + 1), \quad SC \leftarrow 0$
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:	$AC \leftarrow \text{shr } AC, \quad AC(15) \leftarrow E, \quad E \leftarrow AC(0)$
CIL	rB_6:	$AC \leftarrow \text{shl } AC, \quad AC(0) \leftarrow E, \quad E \leftarrow AC(15)$
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}:	$AC(0\text{–}7) \leftarrow INPR, \quad FGI \leftarrow 0$
OUT	pB_{10}:	$OUTR \leftarrow AC(0\text{–}7), \quad FGO \leftarrow 0$
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:	$IEN \leftarrow 0$

[http://www.ee.oulu.fi/research/tklab/courses/521415A/exercises/]