Computer Architecture I

CEG2136

Review 2021

Older Final Exams

uOttawa

L'Université canadienne
Canada's university

Université d'Ottawa | University of Ottawa

www.uOttawa.ca

Q1.1 For 1 K memory locations you need:

(a) 8 address lines

(b) 10 address lines

(c) 12 address lines

(d) None of the above

A memory with a capacity of 2^m words requires m address lines.
So, for $2^m = 2^{10}$, we need $m = 10$ address lines

Address m bits	Location = N bit word
0000	WORD 0
0001	WORD 1
0010	WORD 2
0011	WORD 3
...	...
$(2^m\text{-}1)_2$	WORD 2^m-1

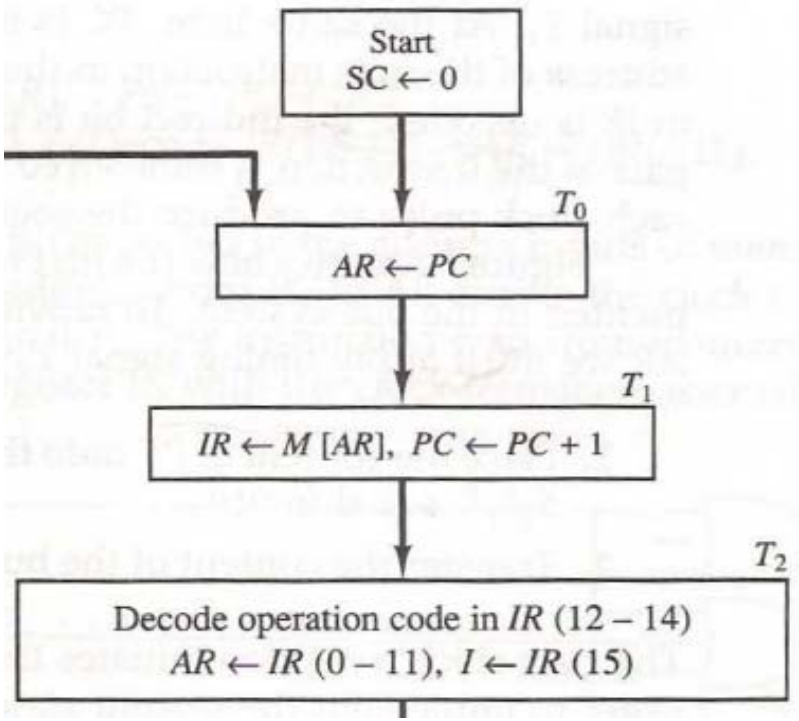
Q1.2 Which CPU register provides the address from which the next instruction opcode is to be fetched?

(a) Instruction register IR

(b) Accumulator AC

(c) Program counter PC

(d) None of these



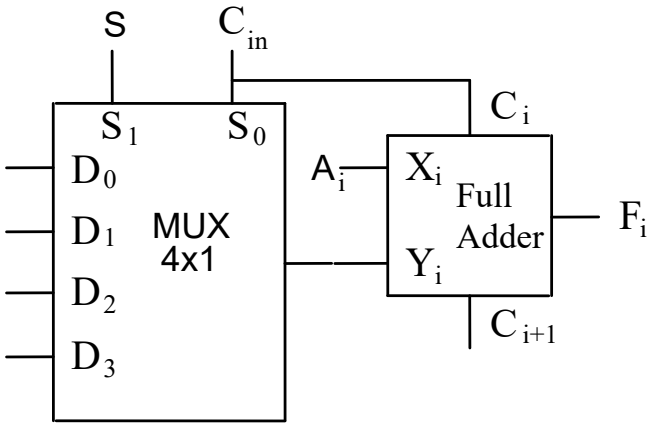
Q1.4 An arithmetic unit performs basic operations on two numbers A and B, under the control of two bits S and C_{in}., as shown in the following table:

S	Cin = 0	Cin = 1
0	F = A + B (addition)	F = A + 1 (increment A)
1	F = A - 1 (decrement A)	F = A + B'+ 1 (subtraction)

S	Cin = 0	Cin = 1
0	$F = A_2\,A_1\,A_0 + B_2\,B_1\,B_0$	$F = A_2\,A_1\,A_0 + 0\ \ 0\ \ 0\ \ +1$
1	$F = A_2\,A_1\,A_0 + 1\ \ 1\ \ 1$	$F = A_2\,A_1\,A_0 + B'_2\,B'_1\,B'_0 + 1$

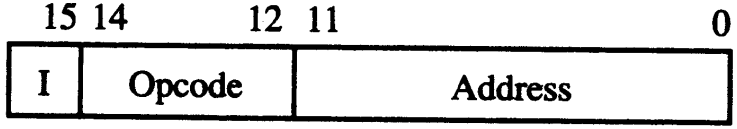
$S\ C_{in}$ $S_1\ S_0$	X_i	Y_i
00	A_i	B_i
01	A_i	0
10	A_i	1
11	A_i	B'_i

The following combinational circuit (multiplexor and full adder) is used to implement the functions described above for bit *i*.
Select from the following combinations of multiplexor inputs which set of logic values implements correctly the above functions



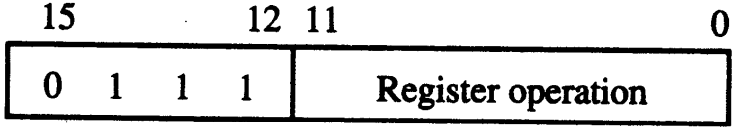
(a) $D_0=B_i'$, $D_1=0$, $D_2=1$, $D_3=B_i$
(b) $D_0=0$, $D_1=B_i$, $D_2=B_i'$, $D_3=1$
(c) $D_0=1$, $D_1=B_i'$, $D_2=B_i$, $D_3=0$
(d) $D_0=B_i$, $D_1=0$, $D_2=1$, $D_3=B_i'$
(e) $D_0=1$, $D_1=B_i$, $D_2=B_i'$, $D_3=0$

Q2. The block diagram of the basic computer that was introduced in chapter 5 of Mano's textbook is given in the annex, along with its instruction list. The instruction word is 16 bit long and has the following structure …



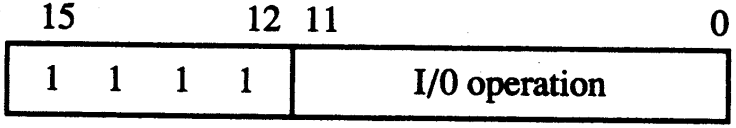
(Opcode = 000 through 110)

(a) Memory – reference instruction



(Opcode = 111, $I = 0$)

(b) Register – reference instruction



(Opcode = 111, $I = 1$)

(c) Input – output instruction

Q2.1 At some point, the content of **PC** of the basic computer is **3AF** (all numbers are in hexadecimal) and the content of **AC** is **2EC3**, as shown in the following table. The content of memory is partially given below, as well:

MEMORY

Address	Memory content
3AD	03B5
3AE	ABBA
3AF	93AD
3B0	DEED
3B1	7BEE
3B2	AD08
3B3	10BC
3B4	1CAA
3B5	3B9F
3B6	3BA0

BASIC COMPUTER REGISTERS

	Content before instruction execution
PC	3AF
AC	2EC3
AR	0000
DR	0000
IR	0000
E	0
I	0
SC	

I=0 Direct addr.		I=1 Indirect addr.		
Assembly language syntax	Machine Code (Hex)	Assembly language syntax	Machine Code (Hex)	Description
AND adr	0adr	AND adr i	8(adr)	AND memory word M to AC
ADD adr	1(adr)	ADD adr i	9(adr)	Add memory word M to AC, carry to E
LDA adr	2(adr)	LDA adr i	A(adr)	Load memory word from M to AC
STA adr	3(adr)	STA adr i	B(adr)	Store content of AC in memory M
BUN adr	4(adr)	BUN adr i	C(adr)	Branch unconditionally
BSA adr	5(adr)	BSA adr i	D(adr)	Save return Address in m, Branch to m+1
ISZ adr	6(adr)	ISZ adr i	E(adr)	Increment memory word M & skip if 0
CLA	7800			Clear AC
CLE	7400			Clear E
CMA	7200			Complement AC
CME	7100			Complement E
CIR	7080			Circulate Right E and AC
CIL	7040			Circulate Left E and AC
INC	7020			Increment AC
SPA	7010			Skip next instruction if AC is >0
SNA	7008			Skip next instruction if AC is <0
SZA	7004			Skip next instruction if AC is =0
SZE	7002			Skip next instruction if E is zero
HLT	7001			Halt computer
		INP	F800	Input character to AC & Clear Flag
		OUT	F400	Out character from AC & Clear Flag
		SKI	F200	Skip if Input Flag is on
		SKO	F100	Skip if Output Flag is on
		ION	F080	Turn Interrupt on
		IOF	F040	Turn Interrupt off

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

ADD 3AD I

b. Show the operands and the binary operation that will be performed in the *AC* when the instruction is executed.

2EC3 + 3B9F

c. Fill out the last column ("Content after instruction execution") of the table with 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, all shown at the end of the instruction cycle.

	Content before instruction execution	Content after instruction execution
PC	3AF	3B0
AC	2EC3	6A62
AR	0000	**3B5**
DR	0000	**3B9F**
IR	0000	**93AD**
E	0	0
I	0	1
SC		**000**

ADD 3AD I (Indirect addressing mode!)

AC←AC+M[M(3AD)]

PC←PC+1

AC←AC+DR=2EC3+3B9F

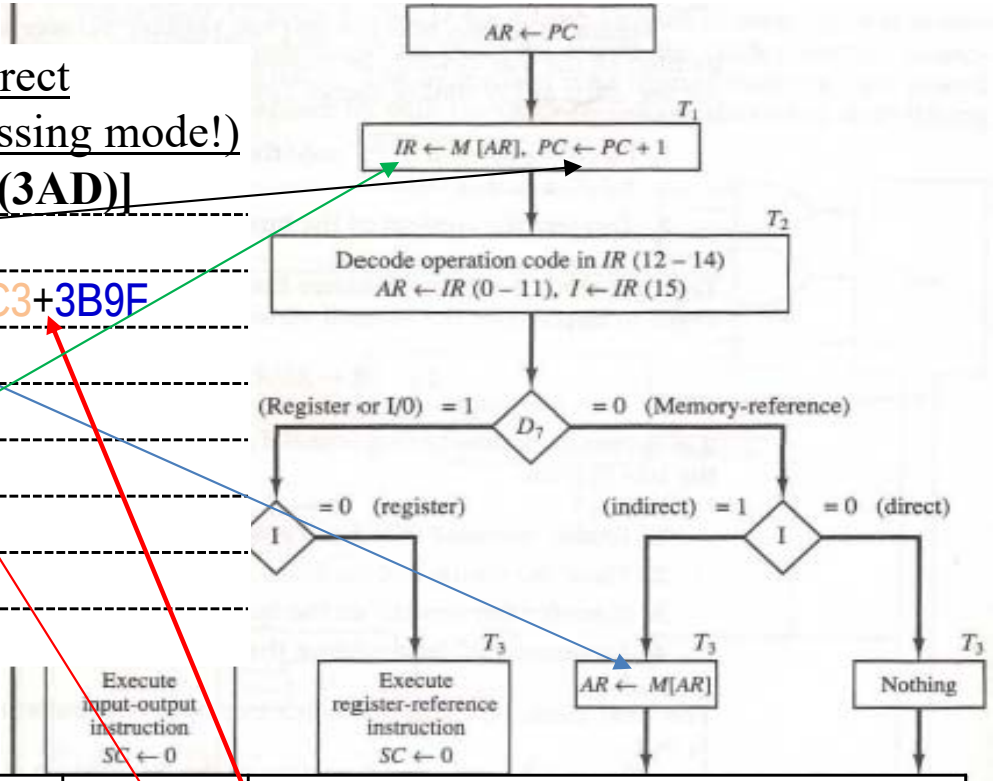
AR←M(3AD)

DR←M(**3B5**)

IR←M(3AF)

E←OFL=0

I←IR(15)



Address	Memory content
3AD	**03B5**
3AE	ABBA
3AF	93AD
3B0	DEED
3B1	7BEE
3B2	AD08
3B3	10BC
3B4	1CAA
3B5	3B9F
3B6	3BA0

AND	D_0T_4:	$DR \leftarrow M[AR]$
	D_0T_5:	$AC \leftarrow AC \wedge DR, SC \leftarrow 0$
ADD	D_1T_4:	$DR \leftarrow M[AR]$
	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:	$M[AR] \leftarrow DR,$
	D_6T_6DR':	$\text{if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$

Q2.2 What is the result, in decimal, of the operation performed by the following assembly program?

	ORG 100	
	LDA OP	**AC=FFA5**
	CMA	**AC=005A**
	INC	**AC=005B**
	ADD OP1	**AC=0094**
	STA OP2	**M(OP2)=0094**
	HLT	
OP1,	0039	
OP,	FFA5	
OP2,	0	**0094**
	END	

005B+
0039
0094

HEX94 = DEC(9 x 16 + 4) = 144 + 4 = 148

I=0 Direct addr.		I=1 Indirect addr.		
Assembly language syntax	Machine Code (Hex)	Assembly language syntax	Machine Code (Hex)	Description
AND adr	0adr	AND adr i	8(adr)	AND memory word M to AC
ADD adr	1(adr)	ADD adr i	**9(adr)**	Add memory word M to AC, carry to E
LDA adr	2(adr)	LDA adr i	A(adr)	Load memory word from M to AC
STA adr	3(adr)	STA adr i	B(adr)	Store content of AC in memory M
BUN adr	4(adr)	BUN adr i	C(adr)	Branch unconditionally
BSA adr	5(adr)	BSA adr i	D(adr)	Save return Address in m, Branch to m+1
ISZ adr	6(adr)	ISZ adr i	E(adr)	Increment memory word M & skip if 0
CLA	7800			Clear AC
CLE	7400			Clear E
CMA	7200			Complement AC
CME	7100			Complement E
CIR	7080			Circulate Right E and AC
CIL	7040			Circulate Left E and AC
INC	7020			Increment AC
SPA	7010			Skip next instruction if AC is >0
SNA	7008			Skip next instruction if AC is <0
SZA	7004			Skip next instruction if AC is =0
SZE	7002			Skip next instruction if E is zero
HLT	7001			Halt computer
		INP	F800	Input character to AC & Clear Flag
		OUT	F400	Out character from AC & Clear Flag
		SKI	F200	Skip if Input Flag is on
		SKO	F100	Skip if Output Flag is on
		ION	F080	Turn Interrupt on
		IOF	F040	Turn Interrupt off

Q2.3 The machine code of this program is stored in a memory of 1 kilo word of 16 bits implemented on an Altera FPGA; give the Quartus .mif file that describes this program

Addr	Memory content		
			ORG 100
100	2107		LDA OP
101	7200		CMA
102	7020		INC
103	1106		ADD OP1
104	3108		STA OP2
105	7001		HLT
106	0039	OP1,	0039
107	FFA5	OP,	FFA5
108	0000	OP2,	0
			END

Addr	+0	+1	+2	+3	+4	+5	+6	+7
100	2107	7200	7020	1106	3108	7001	0039	FFA5
108	0000	0000	0000	0000	0000	0000	0000	0000
110	0000	0000	0000	0000	0000	0000	0000	0000

I=0 Direct addr.		I=1 Indirect addr.		
Assembly language syntax	Machine Code (Hex)	Assembly language syntax	Machine Code (Hex)	Description
AND adr	0adr	AND adr i	8(adr)	AND memory word M to AC
ADD adr	1(adr)	ADD adr i	**9(adr)**	Add memory word M to AC, carry to E
LDA adr	2(adr)	LDA adr i	A(adr)	Load memory word from M to AC
STA adr	3(adr)	STA adr i	B(adr)	Store content of AC in memory M
BUN adr	4(adr)	BUN adr i	C(adr)	Branch unconditionally
BSA adr	5(adr)	BSA adr i	D(adr)	Save return Address in m, Branch to m+1
ISZ adr	6(adr)	ISZ adr i	E(adr)	Increment memory word M & skip if 0
CLA	7800			Clear AC
CLE	7400			Clear E
CMA	7200			Complement AC
CME	7100			Complement E
CIR	7080			Circulate Right E and AC
CIL	7040			Circulate Left E and AC
INC	7020			Increment AC
SPA	7010			Skip next instruction if AC is >0
SNA	7008			Skip next instruction if AC is <0
SZA	7004			Skip next instruction if AC is =0
SZE	7002			Skip next instruction if E is zero
HLT	7001			Halt computer
		INP	F800	Input character to AC & Clear Flag
		OUT	F400	Out character from AC & Clear Flag
		SKI	F200	Skip if Input Flag is on
		SKO	F100	Skip if Output Flag is on
		ION	F080	Turn Interrupt on
		IOF	F040	Turn Interrupt off

Q3. Write a subroutine to subtract two numbers that are stored in memory at 2 consecutive addresses. The address of the minuend is passed to the subroutine through the accumulator AC, while the resulted difference is passed back to the calling program through AC, as well.

Q3.1 Write the subroutine in assembly language using the instruction list of the basic computer. The starting address of the subroutine is HEX 100.

```
ORG 100
DIF  0000          / return address goes here
     STA MIN       /save address of minuend at MIN
     INC           /calculate address of subtrahend
     STA SUB       /save address of subtrahend at SUB
     LDA SUB I     /load subtrahend in AC
     CMA           /find the 2's
     INC           / complement
     ADD MIN I     /add the minuend to the 2's complement of subtrahend
     BUN DIF I     /return to the calling program
MIN, 0000          /address of minuend is stored here by DIF
SUB, 0000          / address of subtrahend is stored here by DIF
```

I=0 Direct	addr.	I=1 Indirect	addr.
Assembly language syntax	Machine Code (Hex)	Assembly language syntax	Machine Code (Hex)
AND adr	0adr	AND adr i	8(adr)
ADD adr	1(adr)	ADD adr i	**9(adr)**
LDA adr	2(adr)	LDA adr i	A(adr)
STA adr	3(adr)	STA adr i	B(adr)
BUN adr	4(adr)	BUN adr i	C(adr)
BSA adr	5(adr)	BSA adr i	D(adr)
ISZ adr	6(adr)	ISZ adr i	E(adr)
CLA	7800		
CLE	7400		
CMA	7200		
CME	7100		
CIR	7080		
CIL	7040		
INC	7020		
SPA	7010		
	7008		
SZA	7004		
SZE	7002		
HLT	7001		
		INP	F800
		OUT	F400
		SKI	F200
		SKO	F100
		ION	F080
		IOF	F040

Q3.2 Write an example in assembly language for a main program that calls this subroutine. This program is stored in the memory starting with address 0.

```
        ORG   0
MAIN    LDA   PTM /load address of the minuend into AC
                  / to pass it to the subroutine DIFF
        BSA   DIF       /call DIF
        STA   RES       /save the result to RES
        HLT
OP1     4321            /minuend
OP2     1234            /subtrahend
RES     0000            / the result will be saved here
PTM     0004            /address of minuend
PTS     0005            /address of subtrahend
```

I=0 Direct	addr.	I=1 Indirect	addr.
Assembly language syntax	Machine Code (Hex)	Assembly language syntax	Machine Code (Hex)
AND adr	0adr	AND adr i	8(adr)
ADD adr	1(adr)	ADD adr i	**9(adr)**
LDA adr	2(adr)	LDA adr i	A(adr)
STA adr	3(adr)	STA adr i	B(adr)
BUN adr	4(adr)	BUN adr i	C(adr)
BSA adr	5(adr)	BSA adr i	D(adr)
ISZ adr	6(adr)	ISZ adr i	E(adr)
CLA	7800		
CLE	7400		
CMA	7200		
CME	7100		
CIR	7080		
CIL	7040		
INC	7020		
SPA	7010		
SNA	7008		
SZA	7004		
SZE	7002		
HLT	7001		
		INP	F800
		OUT	F400
		SKI	F200
		SKO	F100
		ION	F080
		IOF	F040

Q5.2 Use RTL notation to describe the EXECUTION cycle of each of the three newly added instructions; give the corresponding control signals in terms of the instruction code and the sequence counter.

ASR	F020	Arithmetic Right Shift (DR <- DR/2)
DIV	F010	Divide by 4 (DR <- DR/4)
TAT	F008	Swap AC with DR (DR <-> AC)

		RTL							
Instr.	Condition	Micro-operation	Bus Select S_2,S_1,S_0	SHR_{DR}	LD_{DR}	LD_{AC}	INR_{SC}	CLR_{SC}	ALU_{DR}
ASR	$D_7IT_3IR_5$:	DR ← shr DR SC ← 0	000	1	0	0	0	1	0
TAT	$D_7IT_3IR_3$:	AC <= DR, DR <=AC SC ← 0	100	0	1	1	0	1	1
DIV	$D_7IT_3IR_4$:	DR ← shr DR SC ← SC+1	000	1	0	0	1	0	0
	$D_7IT_4IR_4$:	**DR← shr DR SC ← 0**	**000**	**1**	**0**	**0**	**0**	**1**	**0**

NOTE: The term D_7 I T_3 IR_x is obtained from the instruction in IR as follows:

ASR	F	0	2	0
	1111	0000	0010	0000
	I D_7		IR_5	
TAT	F	0	1	0
	1111	0000	0001	0000
	I D_7		IR_4	
DIV	F	0	0	8

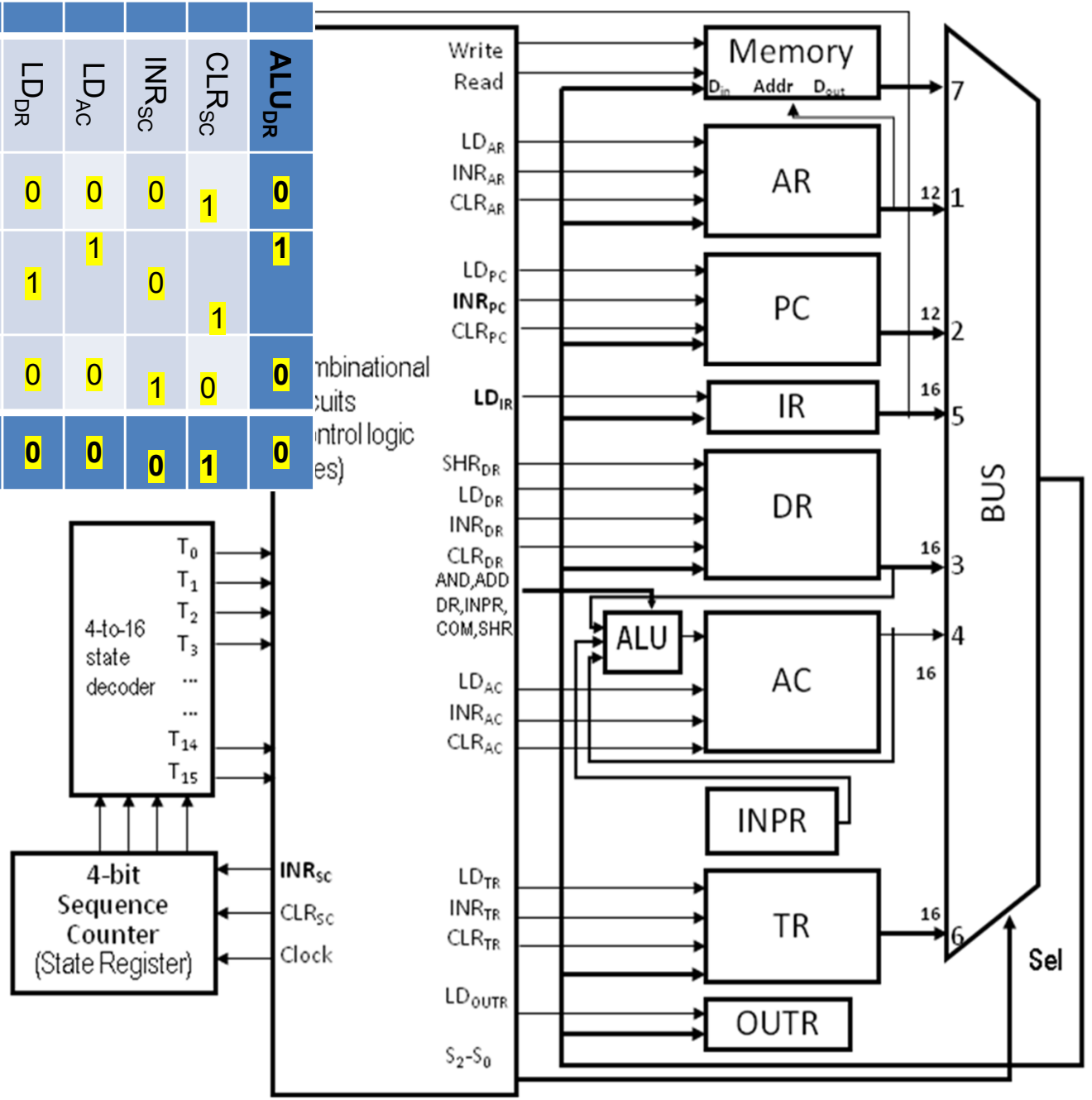
Derive the equations of the control signals from the above table

$SHR_{DR} = D_7IT_3IR_5 + D_7IT_3IR_4 + D_7IT_4IR_4$

$LD_{DR} = D_7IT_3IR_3 = LD_{AC} = ALU_{DR}$

$INR_{SC} = D_7IT_3IR_4$

$CLR_{SC} = D_7IT_3IR_3 + D_7IT_4IR_4 + D_7IT_3IR_5$

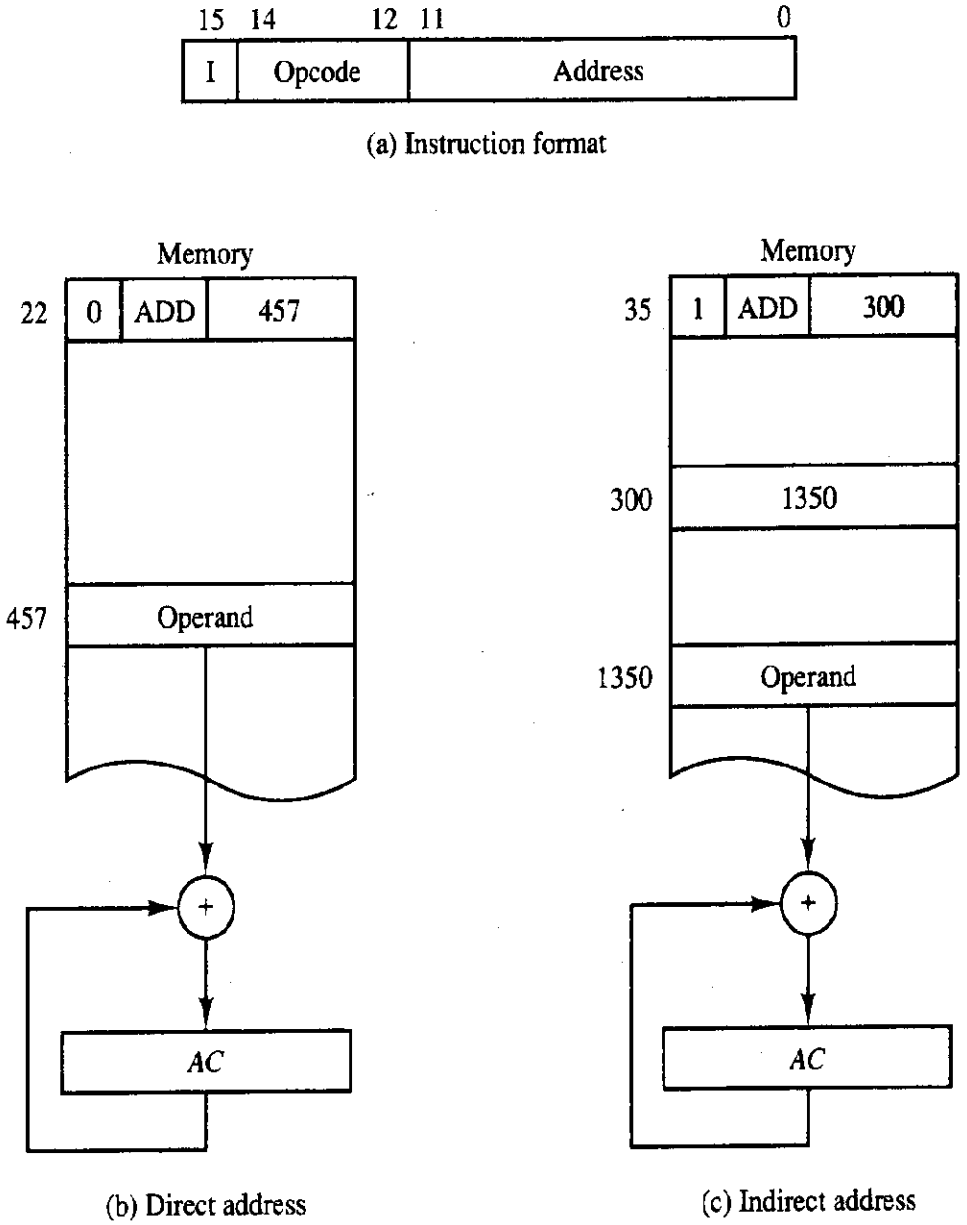


Q1.3 What is the difference between a direct and an indirect address instruction?

- In direct addressing mode, the address field of a memory reference instruction carries a pointer which points to the operand that is stored in memory at the memory location specified by that address field.
- In indirect addressing mode, a pointer points to register (cpu or memory) which contains a pointer to the operand.

How many references to memory are needed for each type of instruction to bring an operand into a processor register, given that the instruction was already fetched into IR?

- direct addressing 1 – direct access to operand
- indirect addressing 2
  1. Read address of operand
  2. Access to operand from just read address



- In **indirect addressing** the Instruction contains address of memory location holding the data address (*pointer*)
- Two level addressing mechanism:
  - 1st level provided by instruction gives *address of memory containing* ***operand's address***
  - 2nd level is the address that specifies where the data (operand) is located