Question 5-1)

Number of words = 256K words = 2^8*2^{10} words
Number of bits pre each word = 32 bit = 2^5 bit
Number of registers = 64 bit = 2^6 Register

a) operation code = 7 bits, register code = 6 bits, address part = 18 bits

b) the instruction word format is as follows:

<table>
<tr><td>I</td><td colspan="7">Operation code</td><td colspan="6">Register part</td><td colspan="18">Address part</td></tr>
<tr><td>31</td><td>30</td><td></td><td></td><td></td><td></td><td></td><td>24</td><td>23</td><td></td><td></td><td></td><td></td><td>18</td><td>17</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td></tr>
</table>

c) data inputs= 32 bits, address inputs=18 bits

Question 5-2)

Direct Address Instruction has its operand address specified directly in the instruction, but the Indirect Address Instruction has the address of the operand's address, i.e. not directly the address of the operand.

Direct Address Instruction: needs 2 visits to memory. The 1st one is to fetch the instruction, 2nd is to bring the operand.

Indirect Address Instruction: needs 3 visits to memory. The 1st one is to fetch the instruction, 2nd is to bring the operand address. And the 3rd is to bring the operand itself.

Question 5-3)

Question	S2	S1	S0	LD of register	Memory	Adder	Solution
A	1	1	1	IR	Read	---	IR <-- M[AR]
B	1	1	0	PC	---	---	PC <-- TR
C	1	0	0	DR	Write	---	DR <- AC,M[AR] <-- AC
d	0	0	0	AC	---	Add	AC <- AC + DR

Question 5-4)

Operation	S_2	S_1	S_0	LD	Memory	Adder/Logic Unit
AR ← PC	0	1	0	AR	-------	-------
IR ← M[AR]	1	1	1	IR	READ	-------
M[AR]←TR	1	1	0	-------	WRITE	-------
AC ←DR, DR←AC (done simultaneously)	1	0	0	DR, AC	-------	Transfer from DR to AC

Question 5-5)

a) We can't use the PC as an address to Memory, because Memory takes its address from AR Register.
We can write it as:

```
AR ← PC
IR ← M[AR]
```

b) There is no such access to AC except for INPR, DR and itself, so the correct form would be:

```
DR ← TR
AC ← AC + DR
```

c) This would change the value of AC to AC + DR. Also the Adder and Logic Unit should save its result in AC not in DR. The correct form would be:

```
AC ← DR, DR ← AC
AC ← AC + DR
AC ← DR, DR ← AC
```

Question 5-6)

Part	Binary instructions	Hexa. code	Operation
A	0001 0000 0010 0100	1024	Add memory word that has the address $024 to AC
B	1011 0001 0010 0100	B124	Store the contents of AC in memory word whose address can be found in memory word having the address $124
C	0111 0000 0010 0000	7020	Increment contents of AC by 1.

Question 5-7)

```
CLE
CME
```

Question 5-9)

Instruction	E	AC	PC	AR	IR
Given	1	A937	021	X	X
CLA	1	0000	022	800	7800
CLE	0	A937	022	400	7400
CMA	1	56C8	022	200	7200
CME	0	A937	022	100	7100
CIR	1	D49B	022	080	7080
CIL	1	526F	022	040	7040
INC	1	A938	022	020	7020
SPA	1	A937	022	010	7010
SNA	1	A937	023	008	7008
SZA	1	A937	022	004	7004
SZE	1	A937	022	002	7002
HLT	1	A937	022	001	7001

Question 5-10)

MRI	IR	AR	DR	AC	PC
Initially	----	----	----	A937	021
AND	0083	083	B8F2	A832	022
ADD	1083	083	B8F2	6229	022
LDA	2083	083	B8F2	B8F2	022
STA	3083	083	----	A937	022
BUN	4083	083	----	A937	083
BSA	5083	083	----	A937	083→084
ISA	6083	083	B8F3	A9F2	022

Question 5-11)

Timing signal	PC	AR	DR	IR	SC
Given	7FF	X	X	X	0
T0	7FF	7FF	X	X	1
T1	800	7FF	X	EA9F	2
T2	800	A9F	X	EA9F	3
T3	800	C35	X	EA9F	4
T4	800	C35	FFFF	EA9F	5
T5	800	C35	0000	EA9F	6
T6	801	C35	0000	EA9F	0

Question 5-12)

a. Instruction is: 932E

b. This instruction is [ADD I 32E], which will add the contents of memory word whose address is stored in memory location 32E (indirect addressing).So AC will have the value 7EC3 + 8B9F = 0A62.

c.

PC	AR	DR	AC	IR	E	I	SC
3B0	9AC	8B9F	0A62	932E	1	1	0101 → 0000

Question 5-13)

Symbol	Opcode	Symbolic designation	Description
XOR	000	AC <---- AC XOR M[EA]	XORing
ADM	001	M[EA]<---- M[EA]+ AC	Adding
SUB	010	AC <== AC-M[EA]	Subtraction
XCH	011	AC<==M[EA], M[EA]<==AC	
SEQ	100	IF(M[EA]==AC) THEN (PC<==PC+1)	Skip on equality
BPA	101	IF (AC>0) THEN (PC <==EA)	Branch if AC>0

Assuming:
Each instr. Begins by T4
Adder can perform XOR directly
Can't do Subtraction

Solution:

First Instruction
D_0T_4: R ← M[AR]
D_0T_5: AC ← AC ⊕ DR, SC←0

Second Instruction:
D_1T_4: DR ← M[AR]
D_1T_5: DR ← AC, AC ← AC + DR
D_1T_6: M[AR] ← AC, AC ← DR, SC ← 0

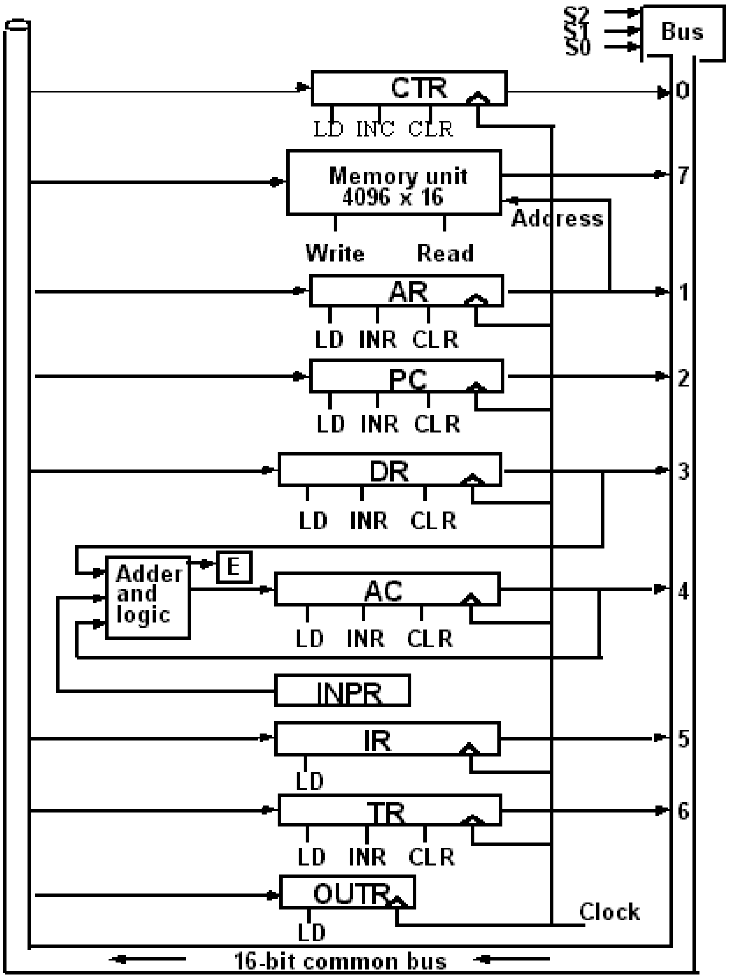
Third Instruction:
D_2T_4: DR ← M[AR]
D_2T_5: DR ← AC, AC ← DR
D_2T_6: AC ← (AC
D_2T_7: AC← AC+1
D_2T_8: AC ← AC + DR, SC ← 0

Fourth Instruction (Exchange):
D_3T4: DR ← M[AR]
D_3T_5: M[AR] ← AC, AC ← DR, SC ← 0

Fifth Instruction:
D_4T_4: DR ← M[AR]
D_4T_5: DR ← AC, AC ← AC ⊕ DR
D_4T_6: if (AC= = 0) then (PC ← PC + 1), AC ← DR, SC ← 0

Sixth instruction (branch if AC>0):
D_5T_4: if (AC(15) == 0 && AC <>0) THEN PC←AR , SC←0

Question 6-14)



We will replace the memory reference instruction ISZ with [LDC address], which will load the CTR register with the value specified by the address.

In addition to that, we will add a new register reference instruction.

ICSZ: increment CTR and skip next instruction if zero.

By using this instruction we don't have to load the memory word into DR, then incrementing DR, and then check it whether it is zero or not, and finally at T_6 we will load the value of DR into memory or increment PC. However, by using ICSZ we will execute this instruction at only one clock cycle at T_4.

Question 6-15)

The memory is to be changed to 65,536 * 16 memory (address of 16 bits)

When I=1 (indirect) there's no change.

But when I=0 (direct address) the address is given by the 16 bits in the next word following the instruction. Modify T_2, $T_{3,}$ and T_4 if necessary to conform to this configuration:

Solution:

56,536*16 memory has 16 address lines (which means 2^{16} words) with each word having 16 btis.

T_0, T_1, and T_2: No change (fetching and decoding)

T_3 D_7: No change (Register or I/O instructions)

Indirect:

ID_7' T_3: AR(12-15) <--- 0

ID_7' T_4: AR <--M[AR]

Direct:

$I'D_7'$ T_3: PC <-- PC + 1

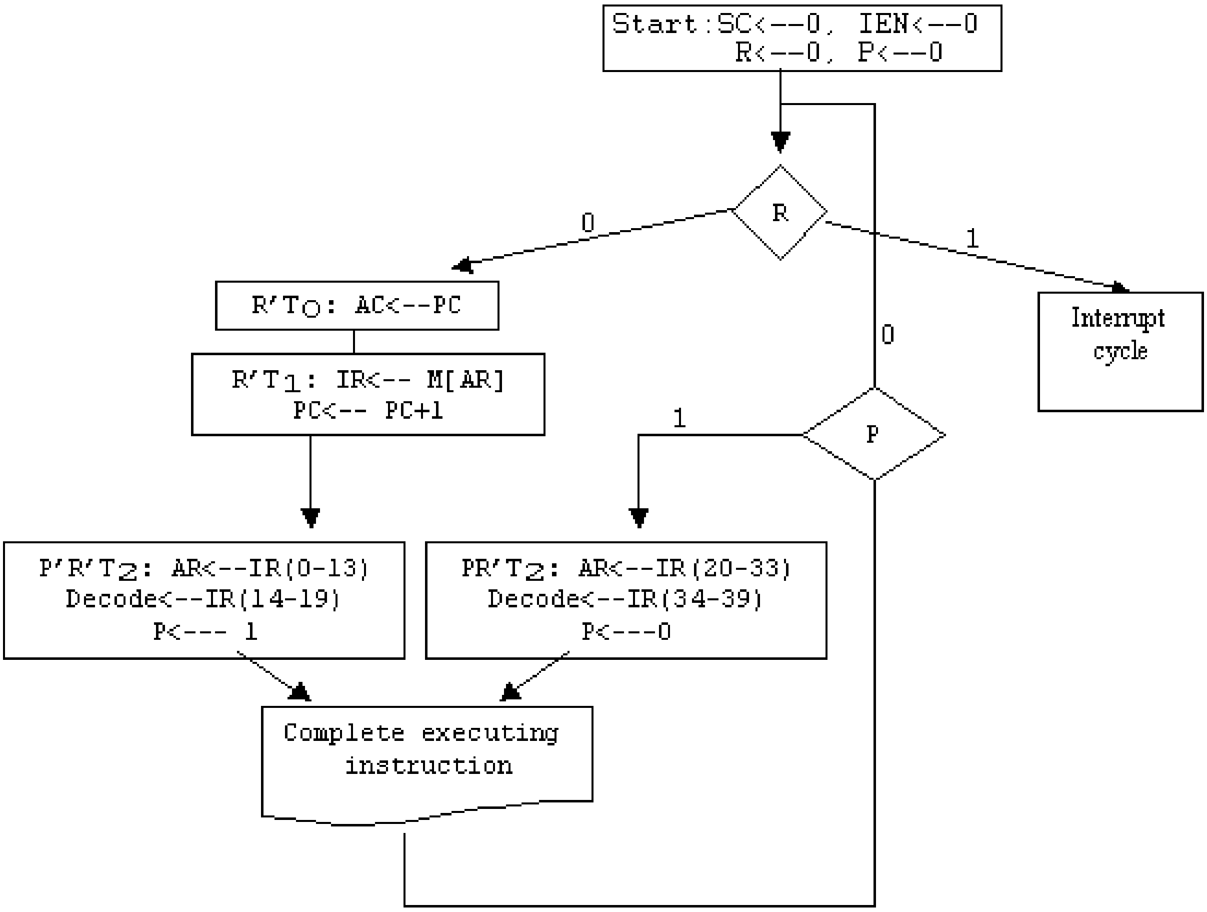
$I'D_7'$ T_4: AR <-- PC

So memory reference instructions would be executed starting from T_5.

Question 5-17)

Capacity of memory = 16,384 word (=2^{14}) with 40 bits per each word. 6 bits/op-code, 14 bits/address-part, where every two instructions are packed into one memory word. And a 40-bit IR register is available in control unit. Formulate fetching and executing instructions?

I will add a flip flop (P) that would do the following:



Question 5-18)

a. The instruction @ address 1 must be a branch to address 300, the address written in the book is wrong (4 bits), so I will consider the first 3 bits of it.

b. The last two instructions would be:

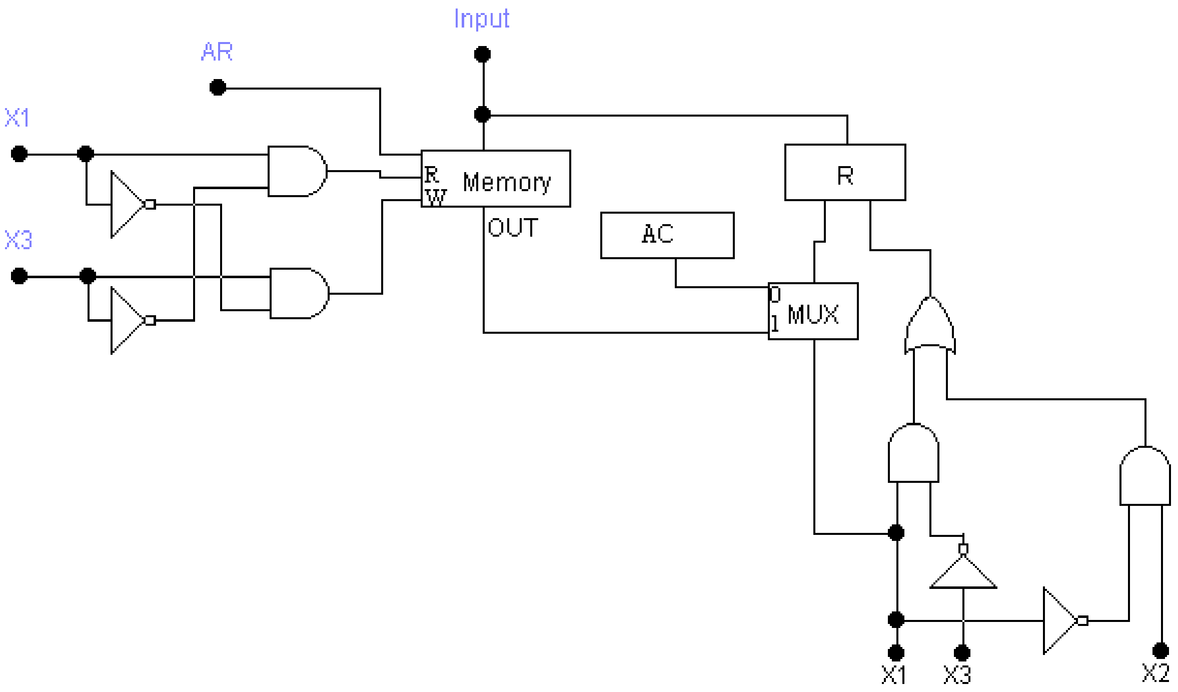
PC←PC+1, IEN←1

The last one would be [BUN I 000]: an indirect branch to memory location 0, and then to reset the SC.

Question 5-19)

$X_3\quad_1$	R<--M[AR]	Read memory word into R
$X_1\quad_2$	R<--AC	Transfer AC to R
$X_1\quad_3$	M[AR]<--R	Write R to memory

The memory has data inputs, data outputs, and control inputs to read and write. Draw the hardware implementation of R and memory?



Question 5-20)

Operations of F flip flop:

XT_3: F🡐1

YT_1: F🡐0

ZT_2: F🡐F’

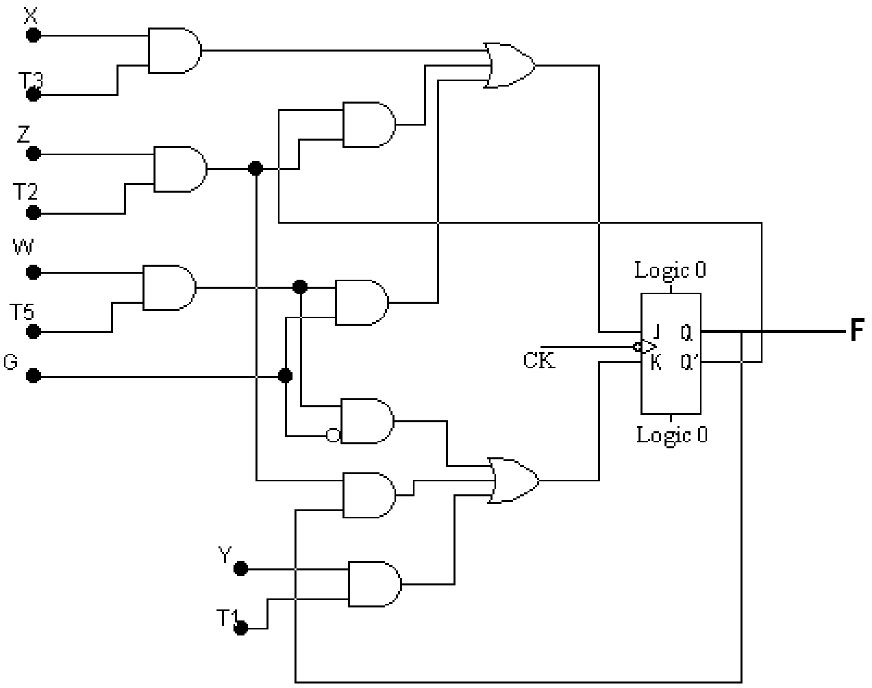
WT_5: F🡐G

PRESET:

J= XT_3 + ZT_2Q' + WT_5G

CLEAR:

K= YT_1 + ZT_2Q + WT_5G'

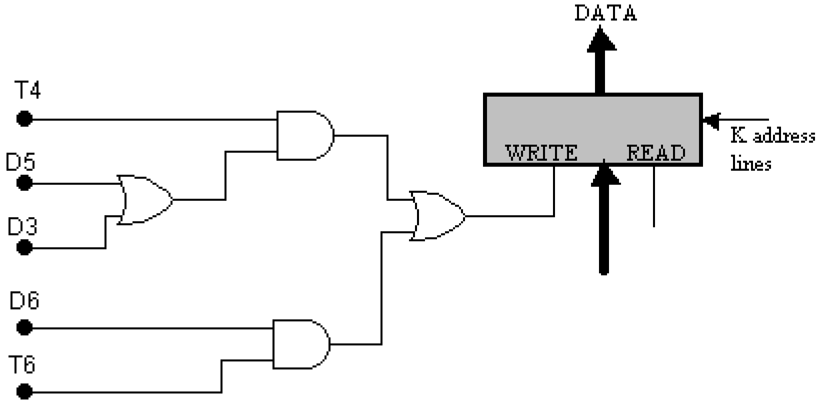


Question 5-21)

Control gates associated with PC:
We assume those signals
A={1:when DR=0 and 0:otherwise}
M=AC(15)
N=1 if AC=0
INC =$R'T_1 + RT_2 + AD_6T_6 + I'D_7T_3[M'B_4 + MB_3 + N'B_2 + E'B_1] + ID_7T_3[B_9(FGI) + B_8(FGO)]$
CLR = RT_1
LD = $D_4T_4 + D_5T_5 +$

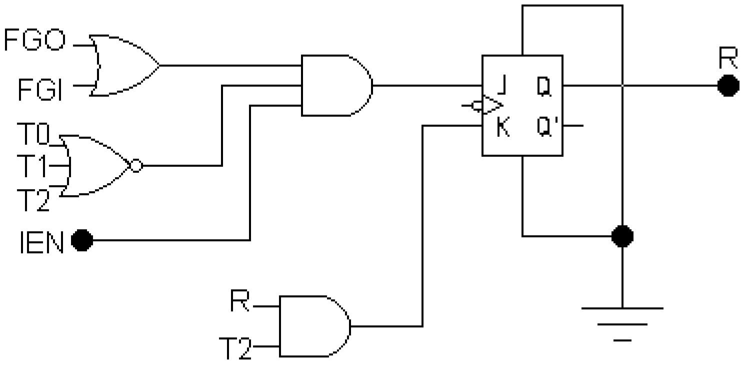
Question 5-22)

Control gates for the write input of the memory:
WRITE = $RT_1 + D_3T_4 + D_5T_4 + D_6T_6$



Question 5-23)

Showing the complete logic of R flip flop in the basic computer using a JK flip flop.
(R=1): $T_0'T_1'T_2'(IEN)(FGI + FGO)$
(R=0): RT_2



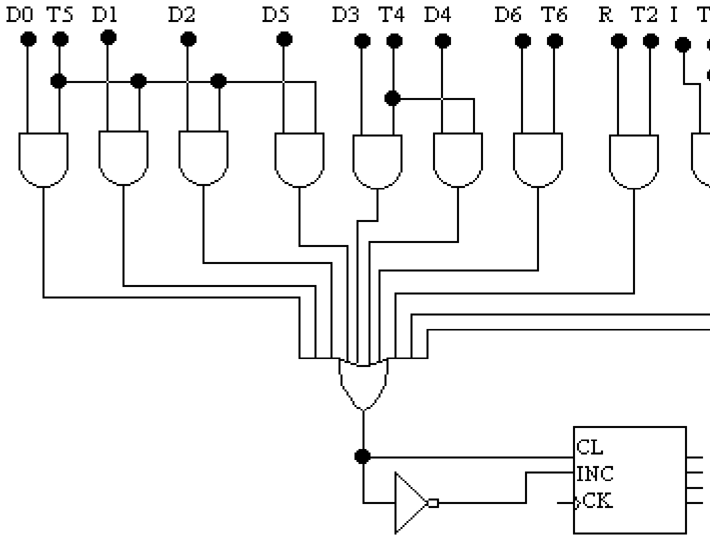
Question 5-24)

$X2 = R'T_O + RT_O + D_5T_4 = T_O + D_5T_4$

Question 5-25)

Deriving the Boolean expression for the gate structure the clears the sequence counter.

$CLR = D_0T_5 + D_1T_5 + D_2T_5 + D_3T_4 + D_4T_4 + D_5T_5 + D_6T_6 + D_7\ I'T_3 + D_7\ IT_3 + R\ T_2$


D0
T5
D1
D2
D5
D3
T4
D4
D6
T6
R
T2
I
T3
D7
I'
CL
INC
CK