COMPUTER ORGANIZATION & ARCHITECTURE

1. **Given R=10, PC=20, and index register X=30 show the value of the accumulator for the following instructions. All memory locations Q contain the value Q+1. Each instruction uses two memory locations.**

   a) LDAC 10
   b) LDAC @10
   c) LDAC R
   d) LDAC (R)
   e) LDAC #10
   f) LDAC $10
   g) LDAC 10(X)

Ans: a) *AC* = 11 b) *AC* = 12 c) *AC* = 10 d) *AC* = 11 e) *AC* = 10 f) *AC* = 33 g) *AC* = 41

2. **Show the code to perform the computation X=A+ (B*C) +D using microprocessors that uses the following instruction formats. Do not modify the values of A, B, C, D. If necessary use temporary location T to store intermediate results**

   a) Three address instruction b) Two address instruction
   c) One address instruction d) Zero address instruction

Ans: a)

```
MUL   R, B, C      R← M [B]*M[C]
ADD   R, A, R      R← M [A] +R
ADD   X, R, D      M[X]← R+M [D]
```

b)

```
MOV   R, B         R←M [B]
MUL   R, C         R←R*M[C]
ADD   R, A         R←R+M [A]
ADD   R, D         R←R+M [D]
MOV   X, R         M[X]←R
```

c)

```
LOAD     B         AC←M [B]
MUL      C         AC←AC*M[C]
ADD      A         AC←AC+M [A]
ADD      D         AC←AC+M [D]
STORE    X         M[X]←AC
```

d)

```
RPN: ABC*+D+
PUSH  A            TOS←A
PUSH  B            TOS←B
PUSH  C            TOS←C
MUL                TOS ←(B*C)
ADD                TOS ←A+ (B*C)
PUSH  D            TOS← D
ADD                TOS← A+ (B*C) +D
POP   X            M[X] ← TOS
```

3. Show the code to perform the computation X=A*B*C+D*(E+F) using microprocessors that uses the following instruction formats. Do not modify the values of A, B, C, D, E, F. If necessary use temporary location T to store intermediate results

a) Three address instruction **b) Two address instruction**

c) One address instruction **d) Zero address instruction**

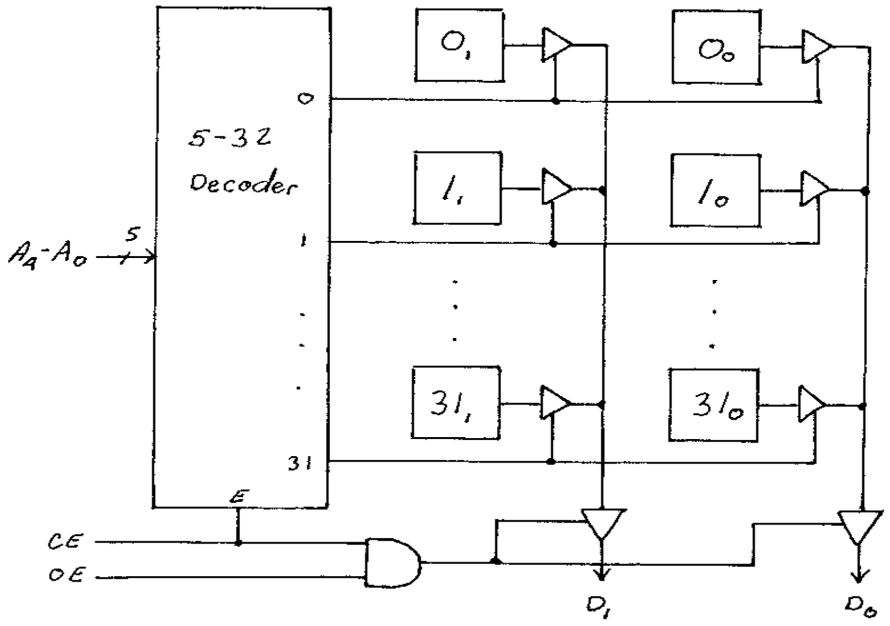
a)
```
MUL   R1, A, B        R1←M[A]*M[B]
MUL   R1, R1, C       R1←R1*M[C]
ADD   R2, E, F        R2←M [E] +M [F]
MUL   R2, D, R2       R2←M [D] *R2
ADD    X, R1, R2      M[X]←R1+R2
```

b)
```
MOV   R1, A           R1←M[A]
MUL   R1, B           R1←R1*M [B]
MUL   R1, C           R1←R1*M[C]
MOV   R2, E           R2←M [E]
ADD   R2, F           R2←R2+M [F]
MUL   R2, D           R2←R2*M [D]
ADD   R1, R2          R1←R1+R2
MOV   X, R1           M[X]←R1
```

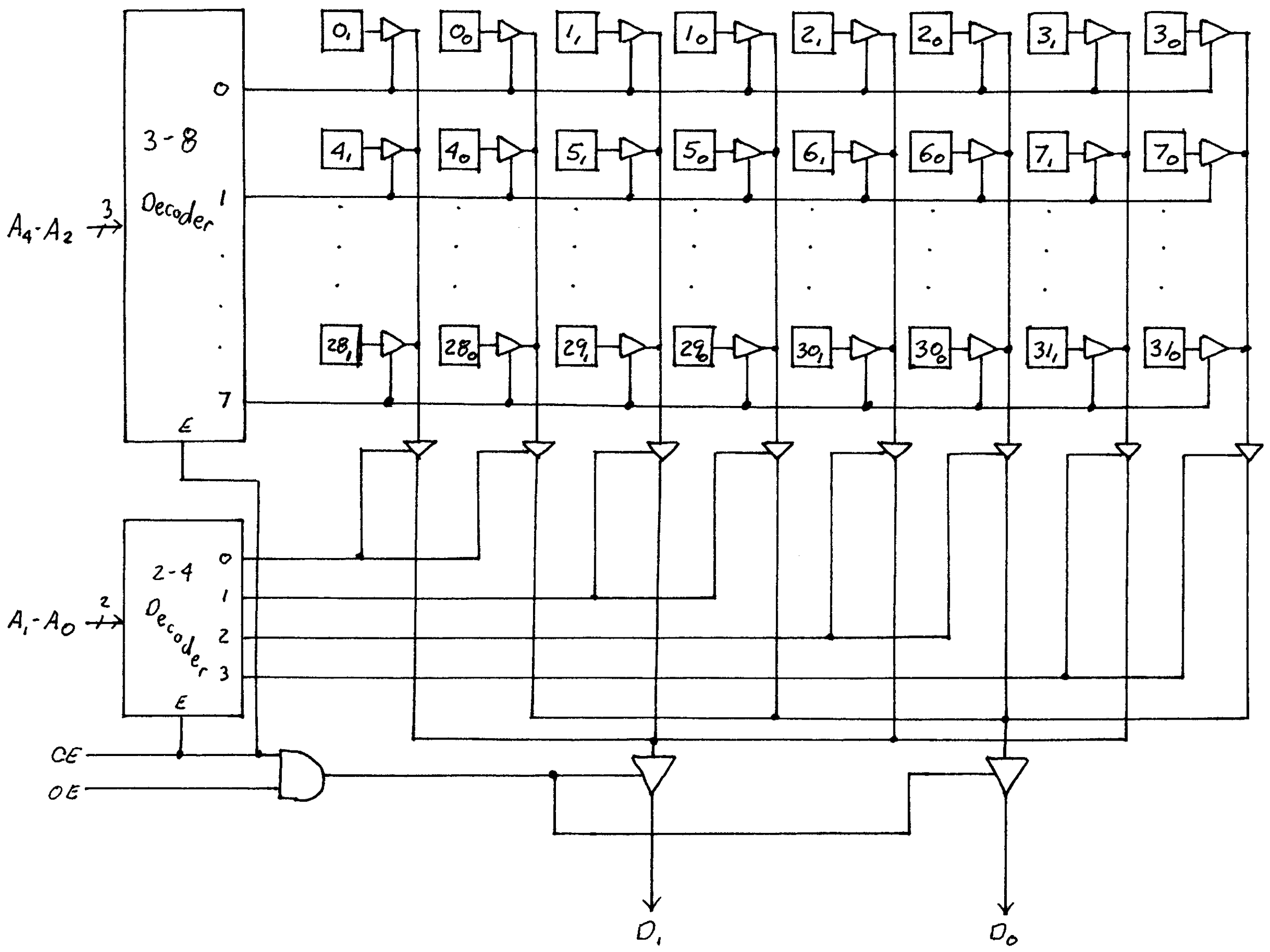
c)
```
LOAD A                AC←M[A]
MUL   B               AC←AC*M [B]
MUL   C               AC←AC*M[C]
STORE T               M [T]←AC
LOAD E                AC←M [E]
ADD F                 AC←AC+M [F]
MULT D                AC←AC*M [D]
ADD   T               AC←AC+M [T]
STORE X               M[X]←AC
```

d) AB*C*DEF+*+
```
PUSH  A               TOS←A
PUSH  B               TOS←B
MUL                   TOS ←(A*B)
PUSH  C               TOS←C
MUL                   TOS ←((A*B)*C)
PUSH D                TOS←D
PUSH E                TOS←E
PUSH F                TOS←F
ADD                   TOS ←(E+F)
MUL                   TOS ←(D*(E+F))
ADD                   TOS ←((A*B)*C) + ((A*B)*C)
POP    X              M[X]←TOS
```

4. Show the internal linear configuration of a 32 X 2 memory chip



5. Show the internal two-dimensional configuration of a 32 X 2 memory chip



6. **Consider the instruction formats of the basic computer. 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**

(a) 0001 0000 0010 0010 = $(1024)_{16}$

ADD $(024)_{16}$ ADD content of M [024] to AC → **ADD 024**

(b) 1 011 0001 0010 0100 = $(B124)_{16}$

I STA (124)6 Store AC in M [M [124]] →**STA @124**

(c) 0111 0000 0010 0000 = $(7020)_{16}$ Register Increment AC → **INC**

7. **Write a program to evaluate the arithmetic statement:** $\boldsymbol{X = \frac{A-B+C*(D*E-F)}{G+H*K}}$

a. Using a general register computer with three address instructions.

b. Using a general register computer with two address instructions.

c. Using an accumulator type computer with one address instructions.

d. Using a stack organized computer with zero-address operation instructions.

a) Three address instructions:

SUB R1, A, B	R1←M [A] - M [B]
MUL R2, D, E	R2←M [D] * M [E]
SUB R2, R2, F	R2← R2 – M [F]
MUL R2, R2, C	R2← R2*M [C]
ADD R1, R1, R2	R1← R1+R2
MUL R3, H, K	R3← M [H] + M [K]
ADD R3, R3, G	R3← R3+ M [G]
DIV X, R1, R3	X← R1/R3

b) Two address instructions:

MOV R1, A	R1←M [A]
SUB R1, B	R1←R1-M [B]
MOV R2, D	R2←M [D]
MUL R2, E	R2←R2*M [E]
SUB R2, F	R2←R2 – M [F]
MUL R2, C	R2←R2*M[C]
ADD R1, R2	R1←R1+R2
MOV R3, H	R3←M [H]
ADD R3, G	R3←R3+M [G]
DIV R1, R3	R1←R1/R3
MOV X, R1	M[X]←R1

c) One Address instructions:

LOAD A	AC← M [A]
SUB B	AC←AC-M [B]
STORE T	M [T]←AC
LOAD D	AC←M [D]
MUL E	AC←AC*M [E]
SUB F	AC←AC-M [F]
MUL C	AC←AC*M[C]
ADD T	AC←AC+M [T]
STORE T	M [T]←AC
LOAD H	AC←M [H]
MUL K	AC←AC*M [K]
ADD G	AC←AC+M [G]
STORE T1	M [T1]←AC
LOAD T	AC←M [T]
DIV T1	AC←AC/M [T1]
STORE X	M[X]←AC

d) Zero address instructions:

RPN: AB-CDE*F-*+GHK*+/

PUSH A	TOS←A
PUSH B	TOS←B
SUB	TOS←(A-B)
PUSH C	TOS←C
PUSH D	TOS←D
PUSH E	TOS←E
MUL	TOS← (D*E)
PUSH F	TOS← F
SUB	TOS←((D*E)-F)
MUL	TOS←C*((D*E)-F)
ADD	TOS←((A-B)+ C*((D*E)-F)
PUSH G	TOS← G
PUSH H	TOS←H
PUSH K	TOS←K
MUL	TOS←(H*K)
ADD	TOS←G+(H*K)
DIV	TOS←((A-B)+ C*((D*E)-F)/(G+(H*K))
POP X	M[X]←TOS

8. **An instruction at address 021 in the basic computer has I = 0, an operation code of the AND instruction, and an address part equal to 083 (all numbers are in hexadecimal). The memory word at address 083 contains the operand B8F2 and the content of AC is A937. Go over the instruction cycle and determine the contents of the following registers at the end of the execute phase: PC, AR, DR, AC, and IR. Repeat the problem six more times starting with an operation code of another memory-reference instruction.**

	PC	AR	DR	AC	IR
Initial	021	—	—	A937	—
AND	022	083	B8F2	A832	0083
ADD	022	083	B8F2	6229	1083
LDA	022	083	B8F2	B8F2	2083
STA	022	083	—	A937	3083
BUN	083	083	—	A937	4083
BSA	084	084	—	A937	5083
ISZ	022	083	B8F3	A937	6083

9. **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.

3AF	**932E**
32E	**09AC**
9AC	**8B9F**

AC = 7EC3

(a) 9 = (1001)

1   001

I=1   ADD

ADD @32E → AC←AC+ M[M[32E]]   7EC3+8B9F

b) AC = 7EC3 (ADD)

DR = 8B9F   0A62   E=1

c) PC = 3AF + 1 = 3BO   IR = 932E

AR = 7AC   E = 1

DR = 8B9F   I = 1

AC = 0A62   SC = 0000

10. **Convert the following numerical arithmetic expression into reverse Polish notation and show the stack operations for evaluating the numerical result.**

(3 + 4)*[10*(2 + 6) + 8]

RPN: 3 4+10 2 6 +*8+*

						6					
					2	2	8		8		
		4		10	10	10	10	80	80	88	
STACK	3	3	7	7	7	7	7	7	7	7	616

OPERATI ON	PUSH(3)	PUSH(4)	ADD	PUSH(10)	PUSH(2)	PUSH(6)	ADD	MUL	PUSH(8)	ADD	MUL

11. **The memory unit of a computer has 256K words of 32 bits each. The computer has an instruction format with four fields: an operation code field, a mode field to specify one of seven addressing modes, a register address field to specify one of 60 processor registers, and a memory address. Specify the instruction format and the number of bits in each field if the in instruction is in one memory word**.

256 K = $2^8 \times 2^{10} = 2^{18}$

op code	Mode	Register	Address
5	3	6	18

= 32

Address	=	18 bits
Mode	=	3 bits
Register	=	6 bits
		27 bits
op code		5 bits
		32 bits

12. A relative mode branch type of instruction is stored in memory at an address equivalent to decimal 750. The branch is made to an address equivalent to decimal 500.

a. What should be the value of the relative address field of the instruction (in decimal)?

b. Determine the relative address value in binary using 12 bits. (Why must the number be in 2's complement?)

c. Determine the binary value in PC after the fetch phase and calculate the binary value of 500. Then show that the binary value in PC plus the relative address calculated in part (b) is equal to the binary value of 500.

(a) Relative address = 500 – 751 = – 251

(b) 251 = 000011111011; – 251 = 111100000101

(c) PC = 751 = 001011101111; 500 = 000111110100

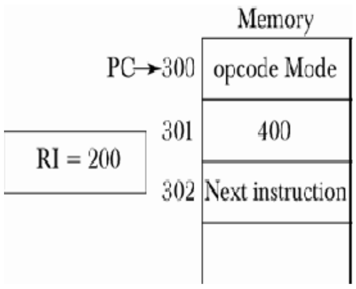
PC = 751 = 001011101111

RA = – 251 = +111100000101

EA = 500 = 000111110100

13. An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is (a) direct; (b) immediate; (c) relative; (d) register indirect; (e) index with R1 as the index register.

(a)direct addressing:
Direct addressing means that the address field contains the address of memory location the instruction is supposed to work with (where an operand "resides").
Effective addres would therefore be 400.



(b) immediate addressing
Immediate addressing means that the address field contains the operand itself.
Effective address would therefore be 301.
(c) relative addressing
Relative addressing means that the address field contains offset to be added to the program counter to address a memory location of the operand.
Effective address would therefore be 302 + 400 = 702.
(d) register indirect addressing
Register indirect addressing means that the address of an operand is in the register. The address field in this case contains just another operand.
Effective address would therefore be in R1 = 200.
(e) indexed addressing with R1 as index register
In indexed absolute addressing the effective address is calculated by taking the contents of the address field and adding the contents of the index register.
Effective address would therefore be 400 + R1 = 400 + 200 = 600.

14. Perform the arithmetic operations below with binary numbers and with negative numbers in signed 2's complement representation. Use seven bits to accommodate each number together with its sign. In each case, determine if there is an overflow by checking the carries into and out of the sign bit position.

a. (+35) + (+40)
b. (-35) + (-40)
c. (-35) - (+40)

When two numbers of n digits each are added and the sum occupies n + 1 digits, we say that an overflow occurred.

A result that contains n + 1 bits cannot be accommodated in a register with a standard length of n bits.

The detection of an overflow after the addition of two binary numbers depends on whether the numbers are considered to be signed or unsigned. When two unsigned numbers are added, an overflow is detected from the end carry out of the most significant position.

In the case of signed numbers, the leftmost bit always represents the sign, and negative numbers are in 2' s complement form. When two signed numbers are added, the sign bit is treated as part of the number and the end carry does not indicate an overflow.

An overflow cannot occur after an addition if one number is positive and the other is negative, since adding a positive number to a negative number produces a result that is smaller than the larger of the two original numbers. An overflow may occur if the two numbers added are both positive or both negative.

An overflow condition can be detected by observing the carry into the sign bit position and the carry out of the sign bit position. If these two carries are not equal, an overflow condition is produced. If the two carries are applied to an exclusive-OR gate, an overflow will be detected when the output of the gate is equal to 1 .

```
a)  +35    0 100011
    +40    0 101000
   ________________
    +75    1 001011
   ________________
```

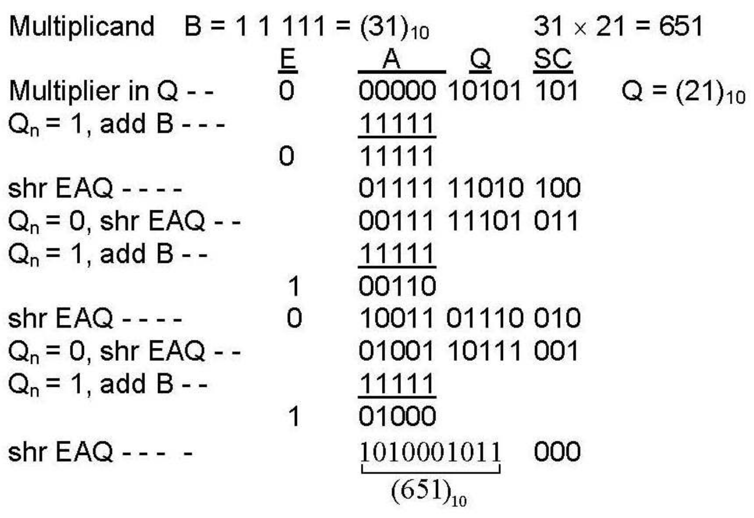
Last two carries are F=0 and E=1. F XOR E= 1.Hence overflow had occurred.

```
b)  -35    1 011101
    -40    1 011000
   ________________
    -75    1 110101
   ________________
```

Last two carries are F=1 and E=0. F XOR E= 1.Hence overflow had occurred.

c) –(35)-(+40) . Add 2's complement of 40 to 2' complement of 35. Same as above

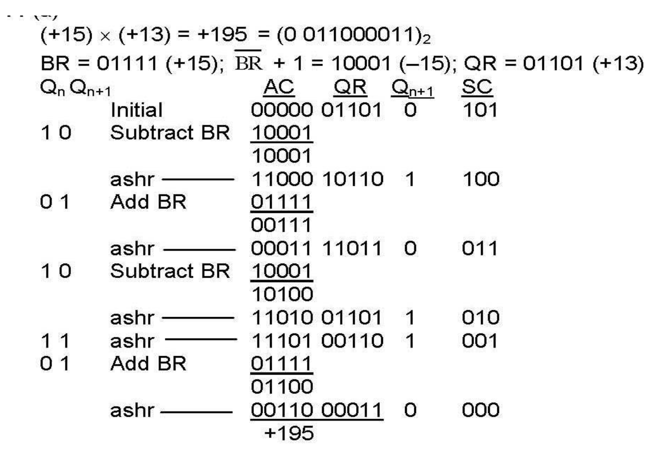
15. **Show the contents of registers E, A, Q, and SC during the process of multiplication of two binary numbers, 11111 (multiplicand) and 10101 (multiplier). The signs are not included**.



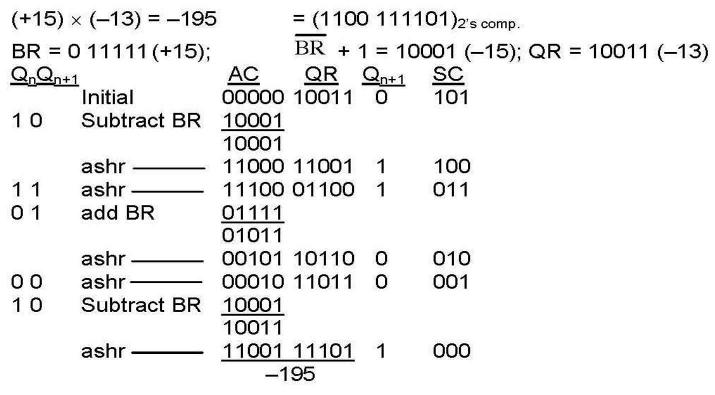
```
Multiplicand    B = 1 1 111 = (31)10          31 × 21 = 651
                         E        A       Q    SC
Multiplier in Q - -      0      00000 10101 101     Q = (21)10
Qn = 1, add B - - -             11111
                         0      11111
shr EAQ - - - -                 01111 11010 100
Qn = 0, shr EAQ - -             00111 11101 011
Qn = 1, add B - -               11111
                         1      00110
shr EAQ - - - -          0      10011 01110 010
Qn = 0, shr EAQ - -             01001 10111 001
Qn = 1, add B - -               11111
                         1      01000
shr EAQ - - -  -                1010001011   000
                                 (651)10
```

16. **Show the step-by-step multiplication process using Booth algorithm when the following binary numbers are multiplied. Assume 5-bit registers that hold signed numbers. The multiplicand in both cases is + 15.**

    a. (+ 15) * (+ 13) **b. (+ 15) * (- 13)**



```
(+15) × (+13) = +195  = (0 011000011)2
BR = 01111 (+15); BR + 1 = 10001 (–15); QR = 01101 (+13)
Qn Qn+1                    AC    QR   Qn+1   SC
        Initial           00000 01101  0     101
1 0     Subtract BR       10001
                          10001
        ashr ———————      11000 10110  1     100
0 1     Add BR            01111
                          00111
        ashr ———————      00011 11011  0     011
1 0     Subtract BR       10001
                          10100
        ashr ———————      11010 01101  1     010
1 1     ashr ———————      11101 00110  1     001
0 1     Add BR            01111
                          01100
        ashr ———————      00110 00011  0     000
                            +195
```



```
(+15) × (–13) = –195            = (1100 111101)2's comp.
BR = 0 11111 (+15);             BR + 1 = 10001 (–15); QR = 10011 (–13)
QnQn+1                     AC    QR   Qn+1   SC
        Initial           00000 10011  0     101
1 0     Subtract BR       10001
                          10001
        ashr ———————      11000 11001  1     100
1 1     ashr ———————      11100 01100  1     011
0 1     add BR            01111
                          01011
        ashr ———————      00101 10110  0     010
0 0     ashr ———————      00010 11011  0     001
1 0     Subtract BR       10001
                          10011
        ashr ———————      11001 11101  1     000
                               –195
```

17. Show the contents of registers E, A, Q, and SC during the process of division of

(a) 10100011 by 1011; (b) 00001111 by 0011. (Use a dividend of eight bits.)

a)

$$\frac{10100011}{1011} = 1110 + \frac{1001}{1011} \qquad \frac{163}{11} = 14 + \frac{9}{11}$$

$B = 1011$ $\quad \bar{B} + 1 = 0101$ $\quad DVF = 0$

	E	A	Q	SC
Dividend in AQ - - - -	0	1010	0011	100
shl EAQ - - - - -	1	0100	0110	
add $\bar{B}$ + 1, suppress carry - -		0101		
E = 1, set Q_n to 1 - - - -	1	1001	0111	011
shl EAQ - - - - - -	1	0010	1110	
add $\bar{B}$ + 1, suppress carry -		0101		
E = 1, set Q_n to 1 - - - -	1	0111	1111	010
shl EAQ - - - - - -	0	1111	1110	
add $\bar{B}$ + 1, carry to E - -		0101		
E = 1, set Q_n to 1 - - - -	1	0100	1111	001
shl EAQ - - - - - -	0	1001	1110	
add $\bar{B}$ + 1, carry to E - - -		0101		
E = 0, leave Q_n = 0 - - - -	0	1110	1110	
add B - - - - - - -		1011		
restore remainder - -	1	$\underbrace{1001}_{\text{remainder}}$	$\underbrace{1110}_{\text{quotient}}$	000

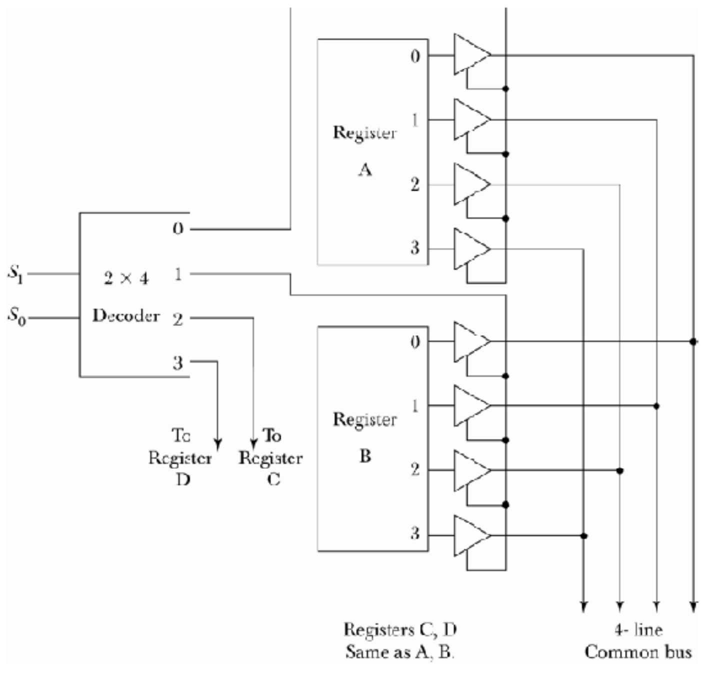
b)

$$\frac{1111}{0011} = 0101$$

$B = 0011$ $\quad \bar{B} + 1 = 1101$

	E	A	Q	SC
Dividend in Q, A = 0 - - - -		0000	1111	100
shl EAQ - - - - - -	0	0001	1110	
add $\bar{B}$ + 1 - - - -		1101		
E = 0, leave Q_n = 0 - - - -	0	1110	1110	
add B - - - - - -		0011		
restore partial remainder - -	1	0001		011
shl EAQ - - - - - -	0	0011	1100	
add $\bar{B}$ + 1 - - - - -		1101		
E = 1, set Q_n to 1 - - - - -	1	0000	1101	010
shl EAQ - - - - - - -	0	0001	1010	
add $\bar{B}$ + 1 - - - - -		1101		
E = 0, leave Q_n = 0 - - - - -	0	1110	1010	
add B - - - - - -		0011		
restore partial remainder - - -	1	0001		001
shl EAQ - - - - - - - -	0	0011	0100	
add $\bar{B}$ + 1 - - - - - -	1101			
E = 1, set Q_n to 1 - - - - -	1	$\underbrace{0000}_{\text{remainder}}$	$\underbrace{0101}_{\text{quotient}}$	000

18. Draw a common bus system using three state buffer and decoders



19. A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers.

a. How many selection inputs are there in each multiplexer?

b. What size of multiplexers are needed?

c. How many multiplexers are there in the bus?

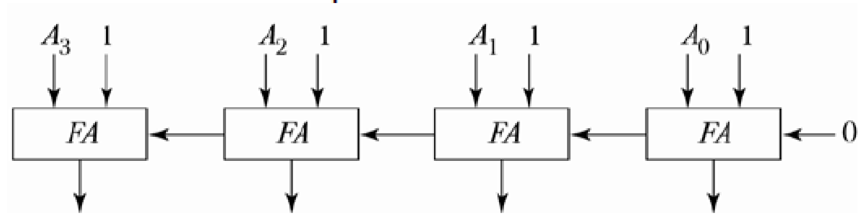
(a) 4 selection lines to select one of 16 registers.

(b) 16 × 1 multiplexers.

(c) 32 multiplexers, one for each bit of the registers.

20. Design a 4-bit combinational circuit decrementer using four full-adder circuits.

A – 1 = A + 2's complement of 1 = A + 1111



21. The 8-bit registers AR, BR, CR, and DR initially have the following values:

AR = 11110010 BR = 11111111 CR = 10111001 DR = 11101010

Determine the 8-bit values in each register after the execution of the following sequence of microoperations.

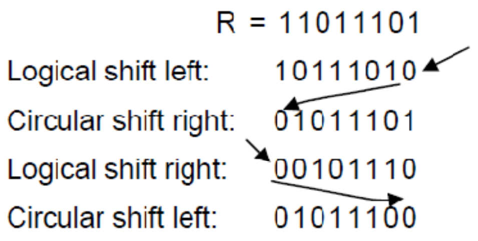
AR ←AR + BR	**Add BR to AR**
CR ←CR ^ DR, BR ←BR + 1	**AND DR to CR, increment BR**
AR ←AR – CR	**Subtract CR from AR**

```
(a)   AR = 11110010
      BR = 11111111(+)
      AR = 11110001            BR = 11111111   CR = 10111001   DR= 1110
1010

(b)   CR = 10111001            BR = 1111 1111
      DR = 11101010(AND)                   +1
      CR = 10101000            BR = 0000 0000   AR = 1111 0001 DR = 11101010


(c)   AR = 11110001 (−1)
      CR = 10101000
      AR = 01001001; BR = 00000000; CR = 10101000;        DR = 11101010
```

22. An 8-bit register contains the binary value 10011100. What is the register value after an arithmetic shift right? Starting from the initial number 10011100, determine the register value after an arithmetic shift left, and state whether there is an overflow.

R = 10011100
Arithmetic shift right: 11001110
Arithmetic shift left: 00111000 overflow because a negative number changed to positive.

23. Starting from an initial value of R=11011101, determine the sequence of binary values in R after a logical shift-left, followed by a circular shift-right, followed by a logical shift-right and a circular shift-left.



R = 11011101

Logical shift left:	10111010
Circular shift right:	01011101
Logical shift right:	00101110
Circular shift left:	01011100

24. a.How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 bytes?

b.How many lines of the address bus must be used to access 2048 bytes of memory? How many of these lines will be common to all chips?

c. How many lines must be decoded for chip select? Specify the size of the decoder.

(a) $\frac{2048}{128} = 16\,\text{chips}$

(b) $2048 = 2^{11}$ 11 lines to address 2078 bytes.

$128 = 2^{7}$ $\underline{\ 7}$ lines to address each chip

4 lines to decoder for selecting 16 chips

(c) 4 × 16 decoder

25. A computer uses RAM chips of 1024 X 1 capacity

a. How many chips are needed and how should their address lines be connected to provide a memory capacity of 1024 bytes?

b. How many chips are needed to provide a memory capacity of 16K bytes? Explain in words how the chips are to be connected to the address bus.

(a) 8 chips are needed with address lines connected in parallel.

(b) 16 × 8 = 128 chips. Use 14 address lines (16 k = 2^{14})

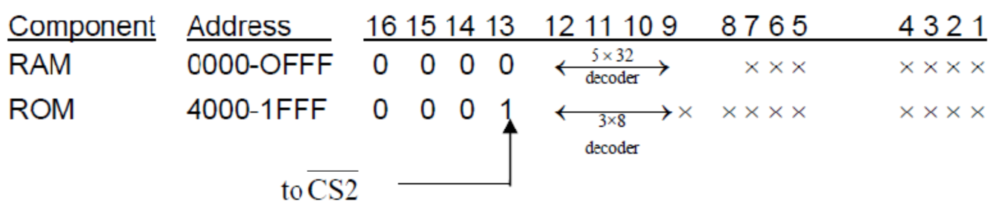
10 lines specify the chip address

4 lines are decoded into 16 chip-select inputs.

26. A memory system of 4096 bytes of RAM and 4096 bytes of ROM is constructed from 128X8 RAM chips and 512 X 8 ROM chips. List the memory address map and indicate what size decoders are needed.

4096/128 = 32 RAM chips; 4096/512 = 8 ROM chips.

4096 = 2^{12} – There 12 common address lines +1 line to select between RAM and ROM.



Component	Address	16 15 14 13	12 11 10 9	8 7 6 5	4 3 2 1
RAM	0000-OFFF	0 0 0 0	$\xleftrightarrow[\text{decoder}]{5\times32}$	× × ×	× × × ×
ROM	4000-1FFF	0 0 0 1	$\xleftrightarrow[\text{decoder}]{3\times8}$ ×	× × × ×	× × × ×

to $\overline{CS2}$

27. The access time of a cache memory is 100 ns and that of main memory is 1000 ns. It is estimated that 80% of the memory requests are for read and the remaining 20% are for write. The hit ratio for read accesses only is 0.9. A write-through procedure is used.

(a) What is the average access time of the system considering only memory read cycles?

(b) What is the average access time of the system for both read and write requests?

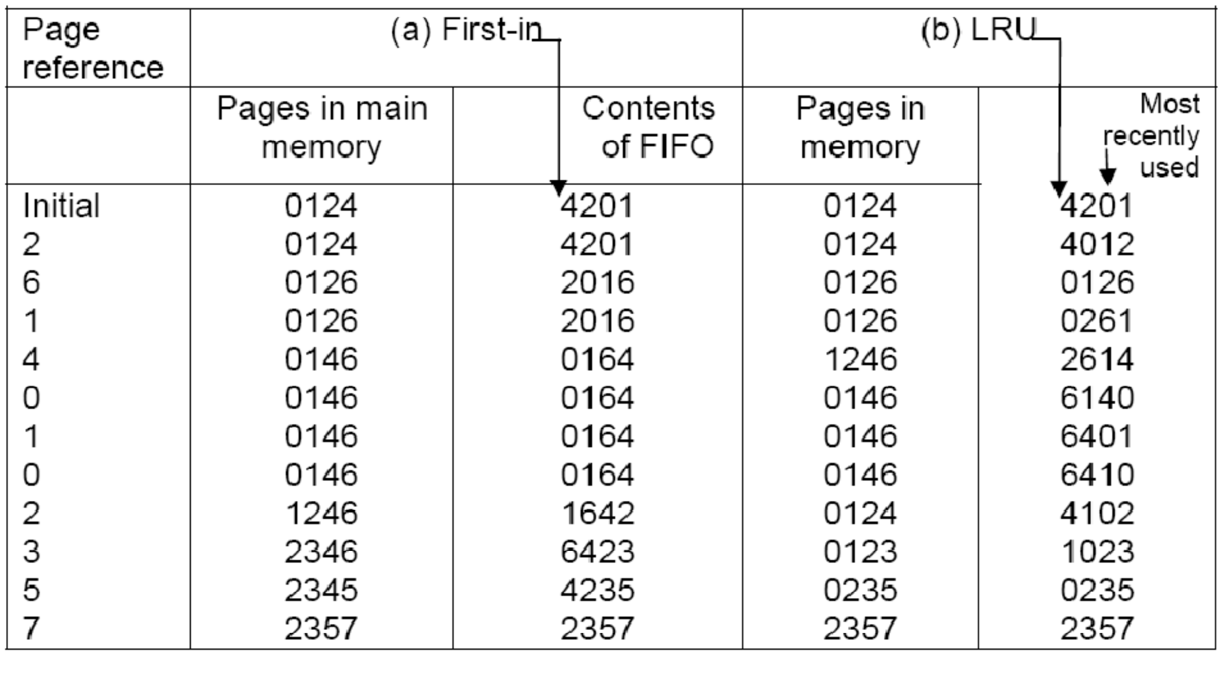
a)Average access time for memory read in the system is calculated using formula:
average access time read = hitratio x cache access time + (1 - hit_ratio) x main memory access time
soin this case:
average access time read = 0.9 x 100ns + (1 - 0.9) x 1000ns = 90ns x 100ns = 190ns

b) If we take in account both read and write accesses then we have to sum averages for read and write. Read average would take those 80% of overall requests and the average read access time of 190ns we calculated in a) to get 0.8 x 190. Write average would take those 20% of overall requests and the main memory access time of 1000ns to get 0.2 x 1000ns.
Summed together we get: 0.8 x 190ns + 0.2 x 1000ns = 152ns + 200ns = 352ns.

28. A virtual memory system has an address of 8 K words, a memory space of 4K words and page and block sizes of 1K words. The following page reference changes occur during a given time interval. 4 2 0 1 2 6 1 4 0 1 0 2 3 5 7

Determine the four pages that are resident in manin memory after each page reference change if the replacement algorithm used is a) FIFO b) LRU

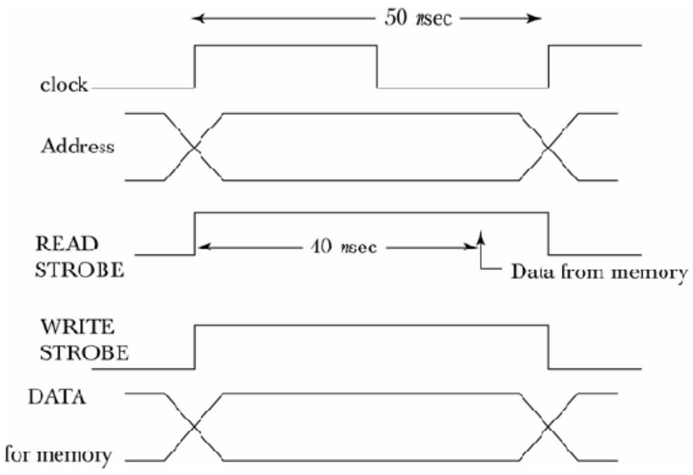
4 2 0 1 2 6 1 4 0 1 0 2 3 5 7



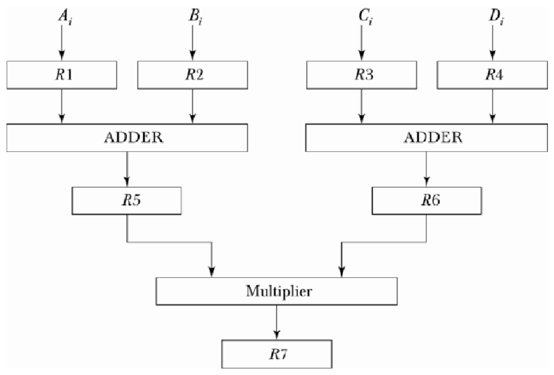
Page reference	(a) First-in		(b) LRU	
	Pages in main memory	Contents of FIFO	Pages in memory	Most recently used
Initial	0124	4201	0124	4201
2	0124	4201	0124	4012
6	0126	2016	0126	0126
1	0126	2016	0126	0261
4	0146	0164	1246	2614
0	0146	0164	0146	6140
1	0146	0164	0146	6401
0	0146	0164	0146	6410
2	1246	1642	0124	4102
3	2346	6423	0123	1023
5	2345	4235	0235	0235
7	2357	2357	2357	2357

29. A CPU with a 20-MHz clock is connected to a memory unit whose access time is 40 ns. Formulate a read and write timing diagrams using a READ strobe and a WRITE strobe. include the address in the timing diagram

$$20\text{ MHz} = 20 \times 10^6\text{ Hz} \qquad T = \frac{10^{-6}}{20} = 50\,\text{n sec.}$$



30. In certain scientific computations it is necessary to perform the arithmetic operation (Ai + Bi)*(Ci + Di) with a stream of numbers. Specify a pipeline configuration to carry out this task. List the contents of all registers in the pipeline for i = 1 through 6



31. Draw a space-time diagram for a six-segment pipeline showing the time it takes to process eight tasks.

Segment	1	2	3	4	5	6	7	8	9	10	11	12	13
1	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8					
2		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8				
3			T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8			
4				T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8		
5					T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	
6						T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8

$(k + n - 1)t_p = 6 + 8 - 1 = 13$ cycles

32. Determine the number of clock cycles that it takes to process 200 tasks in a six-segment pipeline.

k = 6 segments

n = 200 tasks (k + n – 1) = 6 + 200 – 1 = 205 cycles

33. A non pipeline system takes 50 ns to process a task. The same task can be processed in a six-segment pipeline with a clock cycle of 10 ns. Determine the speedup ratio of the pipeline for 100 tasks. What is the maximum speedup that can be achieved?

tn = 50 ns

k = 6

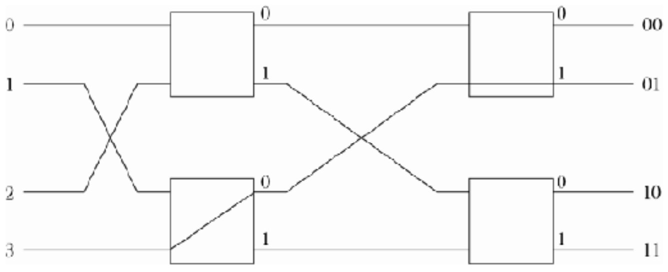
tp = 10 ns

n = 100

$$S = \frac{nt_n}{(k+n-1)t_p} = \frac{100 \times 50}{(6+99) \times 10} = 4.76$$

$$S_{max} = \frac{t_n}{t_p} = \frac{50}{10} = 5$$

34. Construct a diagram for a 4 x 4 omega switching network. Show the switch setting required to connect input 3 to output 1 .



35. Describe the following terminology associated with multiprocessors.

(a) mutual exclusion; (b) critical section; (c) hardware lock; (d) semaphore;

(e) test-and-set instruction.

(a) **Mutual exclusion** implies that each processor claims exclusive control of the resources allocated to it.

(b) **Critical section** is a program sequence that must be completely executed without interruptions by other processors.

(c) **Hardware lock** is a hardware signal to ensure that a memory read is followed by a memory write without interruption from another processor.

(d) **Semaphore** is a variable that indicates the number of processes attempting to use the critical section.

(e) **Test and set instruction** causes a read-modify write memory operation so that the memory location cannot be accessed and modified by another processor.