TUTORIALS:

TUTORIAL 1

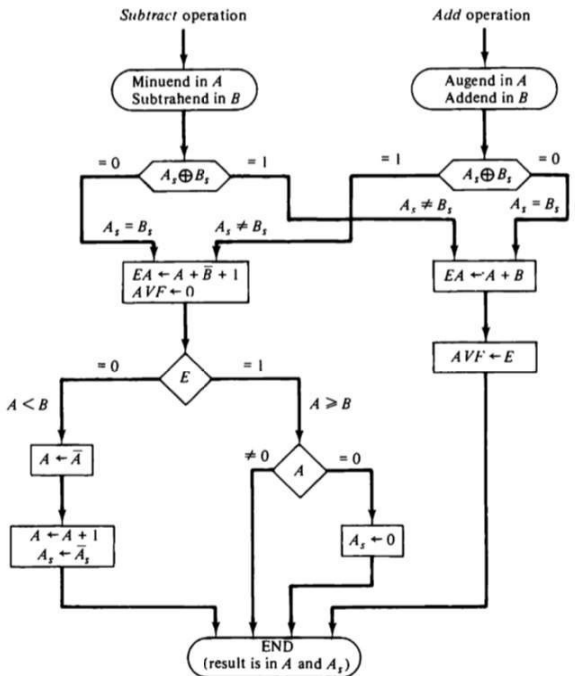
Perform the following operation using signed-magnitude addition subtraction algorithm. The sign bit has been marked in Red.

In each case indicate the value of AVF.

a. **0 101101 + 0 011111**
b. **1 011111 + 1 101101**
c. **0 101101 – 0 011111**
d. **0 101101 – 0 101101**
e. **1 011111 – 0 101101**

Solution:

Follow the flowchart and perform the operations:



Tut 2.

① Multiply $(-9) \times 8$ using Signed Magnitude Multiplication.
② Multiply $(-6) \times (-5)$ using Booth's Multiplication Algo.

① Multiplicand (B) $-9 \Rightarrow$ 1 1001 (1001 → data bit)
Sign bit $B_s = 1$. (1 ↙ Sign bit)

Multiplier (Q) $+8 \Rightarrow$ 0 1000 (1000 → data bit)
Sign bit $Q_s = 0$. (0 ↙ Sign bit)

4 data bits in Multiplier ⇒ Sequence Counter value ⇒ 100.

Sign bit of Multiplication result ⇒ $B_s \oplus Q_s = 1 \oplus 0 = 1$.
So resultant sign bit 1 implies negative sign.

Multiplicand B = 1001.

		E	A	Q	SC
		0	0000	1000	100
$Q_n = 0$	Shr AQ		0000	0100	011
$Q_n = 0$	Shr.		0000	0010	010
$Q_n = 0$	Shr.		0000	0001	001
$Q_n = 1$	~~Shr~~ Add B.		1001		
			1001		
	Shr.		0100	1000	000

Result.

∴ Resultant sign bit was 1.
So Result is $-72 \Rightarrow$ 1 0100 1000 (1 ↙ Sign bit; 0100 1000 → data.)

(2) Multiplicand −6
Multiplier −5

Booth's Algo. is all about 2's complement Multiplication.

6 → 110 5 → 101
1's comp. → 001 1's comp → 010
2's comp. → 010 2's comp → 011

∴ (−6) will be represented in 2's complement as 1010
(−5) will be represented in 2's complement as 1011

∴ Multiplicand (BR) = 1010 } sequence counter
Multiplier (QR) = 1011 } (SC) = 100.

BR = 1010
$\overline{BR}$+1 = 0101 +1 = 0110

Qn	Qn+1	operation	AC	QR	Qn+1	SC.
			0000	1011	0	100
1	0	SUB BR	0110			
			0110			
		ASHR	0011	0101	1	011
1	1	ASHR	0001	1010	1	010
0	1	ADD BR	1010			
			1011			
		ASHR	1101	1101	0	001
1	0	SUB BR	0110			
			(1) 0011			
			↓ discard			
		ASHR	00011110		1	000

⇓
Result.
MSB = 0 so positive result.
Result is 00011110 ⇒ +30.

Tut 3:

1. a. Memory capacity 2048 bytes.
one RAM chip size 128 X 8
So Reqd. no. of RAM chip = $\frac{2048}{128}$ = 16 chips.

b. $2048 = 2^{11}$ ∴ 11 address bus lines should be used to access 2048 bytes.
each RAM chip of size 128 X 8 ⇒ $128 = 2^7$.
so 7 address lines will be common to ~~each~~ all chips.

c. total no. of lines of address bus = 11.
common no. to ~~ea~~ all the chips 7.
So 4 lines must be decoded for chip select
there are 16 chips.
So 4 X 16 decoder needs to be used.

2. The cache consists of 64 lines.
64 lines are divided into four line sets.
So, in 1 set there are four-lines.
∴ 4 lines in 1 set.
So 64 lines in $\frac{1}{4} \times 64 = 16$ sets.
So the cache is divided into 16 sets of 4 lines each.
So 4 bits are required to identify the set no.
Now main memory consists of 4K blocks = $2^2 \cdot 2^{10}$
$= 2^{12}$ blocks
So total address lines = 12.
4 bits are required to identify set no.
So length of tag field = 12 - 4 = 8.
each word length $128 = 2^7$.
So 7 bits are required to specify the word.

∴ Main memory address =

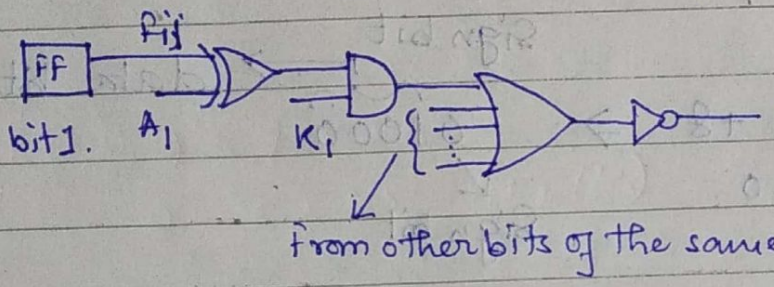
Tag	set	word
8	4	7

3. Match logic for one word

$$M_i = \prod_{j=1}^{n} (A_j F_{ij} + A_j' F_{ij}' + K_j')$$

$$= \prod_{j=1}^{n} [(A_j \oplus F_{ij})' + K_j']$$

So, $M_i' = \sum_{j=1}^{n} (A_j \oplus F_{ij}) \cdot K_j$ [Match logic complement for all the words of Associative Memory



Tutorial - 4.

Addressing Mode	Effective Address	Content of AC.
Immediate operand	201.	500
Direct	500	800
Indirect	800	300
Relative	702	325
Indexed	600	900
Register	~~400~~ —	~~325~~ 400
Register Indirect	400	700.

/ Two address instruction.

/ immediate addressing → effective address → ~~50~~ 201.
content → 500.

/ Direct → check content of location 500 → 800

/ Indirect → check content of 800 → 300.

/ Relative address → In the indirect mode the effective addre
is stored in memory at address 500.
So in relative mode effective address is
A + PC = 500 + 202 = 702
content of 702 → 325

/ Indexed address → A + XR = 500 + 100 = 600.
↓
XR is given as index register content.

/ Register → In the register mode the operand is in Register ~~mode~~ R1. so the ~~operand~~ content loaded into AC is 400. No effective address as directly loaded from regi.

/ Register Indirect → In register indirect effective address will 400 so, content of AC will be 700.

Tut 5: Q1.

Memory unit has 256K words of 32 bits.

$\therefore$ 256 K words $\Rightarrow 2^8 \cdot 2^{10} = 2^{18}$

So 18 address bits.

apart from this a mode field to specify one of seven addressing modes.

$\therefore$ to specify seven addressing mode we need 3 bits as $2^3 = 8$.

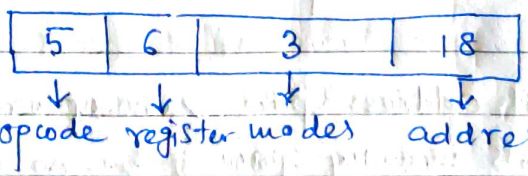
So for Mode $\rightarrow$ 3 bits.

60 processor registers are there

so corresponding required bits are 6. as $2^6 = 64$

So out of 32 bits, 18 bits were used for address.
3 bits are used for specifying modes
6 bits are " " " register.
total of 27 bits
remaining bits are $32 - 27 = 5$ bits.

$\therefore$ structure



Tut 5. Q2.

32 bit instructions

12 bit addresses.

So one word, or one address instruction requires 12 bit addresses.

For two word or two address instruction it requires (12+12) bit.

Now instruction is of 32 bits.

opcode	Address 1	Address 2
8 bit	12 bit	12 bit

→ two address instruction = 32 bits.

So there are 8 bit for opcode

total no of possible combination for one and two address instructions can be $2^8 = 256$.

If there are 250 two-address instructions.

so possible no. of one-address instruction = 256 − 250

= 6.

32 bit instruction 12 bit one address instruction

so in case of one address instruction

opcode	Address
6×2^{12}	12 bit.

Maximum no. of one address instruction = $6 \times 2^{12} = 24,576$

Tut 5.

Q3. 24 bits address space ⇒ a. no. of words in address space = 2^{24}.

16 bits memory space ⇒ b. no. of words in memory space = 2^{16}

c. a page consists of 2K words

∴ no of pages = $2^{24}/2K = 2^{24}/2 \cdot 2^{10} = 2^{13}$ pages

no. of blocks = $2^{16}/2K = 2^{16}/2 \cdot 2^{10} = 2^5$

= 32 blocks.

Tut 5.

Q4. Memory unit of $64K \times 16$.

$\Rightarrow 2^6 \cdot 2^{10} \times 16$

$\Rightarrow 2^{16} \times 16$

so 16 bit address, 16 bit data

cache uses direct mapping of block size of four words.

for four words address bit required $2 [2^2 = 4]$.

a. cache memory size = $1K = 2^{10}$.

so total address lines = 10.

for words, address lines = 2.

∴ for block, address lines = $10 - 2 = 8$.

total address lines = 16.

so no. of bits for tag field = $16 - (2+8) = 6$.

so memory structure

tag	block	word
6	8	2

Index = $8+2 = 10$

b. there are 6 bits for tag

1 bit for valid bit

16 bit of data

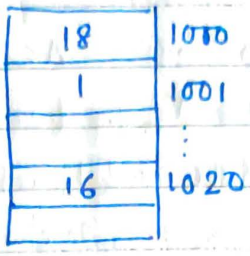
valid	tag	data
1	6	16 = 23

so there are 23 bits in each word of cache.

c. cache can accommodate no. of blocks as $2^8 = 256$.

256 blocks of four words each.

Tut 6. Sol^{n} 1.



1st instruction: MOVI Rs, 1

immediate addressing so, $R_s \leftarrow 1$.

so content of R_s is 1.

2nd instruction: LOAD Rd, 1000 (Rs)

content of displacement addressing so, $R_d \leftarrow [1000 + [R_s]]$

$\Rightarrow R_d \leftarrow [1000 + 1]$

$\Rightarrow R_d \leftarrow [1001]$

$\Rightarrow R_d \leftarrow 1$.

3rd instruction: ADDI Rd, 1000

immediate addressing so, $R_d \leftarrow [R_d] + 1000$

$\Rightarrow R_d \leftarrow 1 + 1000$

$\Rightarrow R_d \leftarrow 1001$.

4th instruction: STOREI 0(Rd), 20

ist displacement addressing, then store

so $0 + [R_d] \leftarrow 20$

$\Rightarrow 0 + 1001 \leftarrow 20$

$\Rightarrow 1001 \leftarrow 20$

so 20 will be moved to 1001 th location.

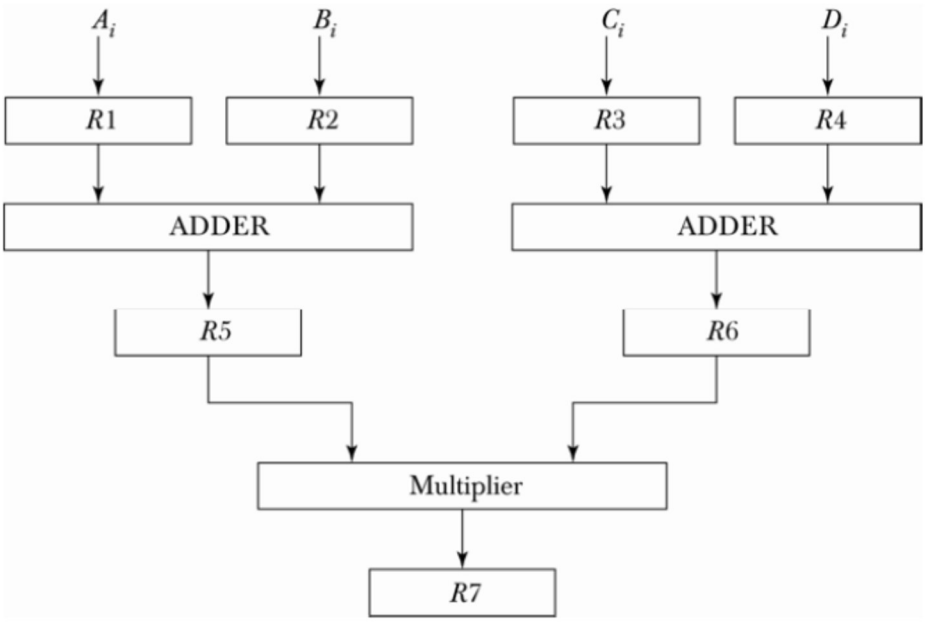
So option (d) is correct.

Tut 6. Sol^{n} 2.

a) Load immediate 20 → AC ← 20.
b) Load direct 20 → AC ← [20] i.e content of 20 i.e 40.
c) Load indirect 20 → AC ← [[20]] i.e content of 40 i.e 60.
d) Load immediate 30, ← AC ← 30.
e) Load direct 30 → AC ← [30] i.e content of 30 i.e 50
f) Load indirect 30 → AC ← [[30]] i.e content of 50 i.e 70.

Solution: Tut 7

1.



2.

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

3.

k = 6 segments
n = 200 tasks $(k + n - 1) = 6 + 200 - 1 = 205$ cycles

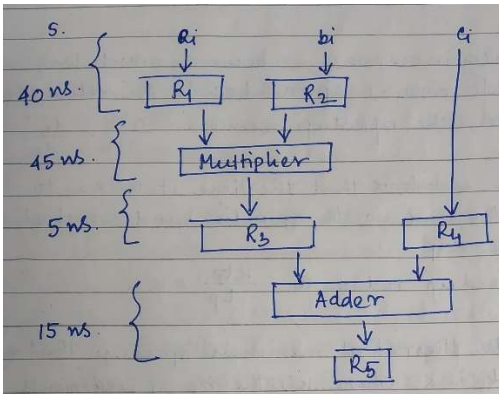
4.

$t_n = 50$ ns
$k = 6$
$t_p = 10$ ns
$n = 100$

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

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

5.



C11.	C12	C13	C14	C15	C16	C17.	C18	C19
T_1	T_2	T_3	T_4	T_5	T_6	T_7		
	T_1	T_2	T_3	T_4	T_5	T_6	T_7	
		T_1	T_2	T_3	T_4	T_5	T_6	T_7

In between timings are not same.

then cycle time is calculated as $t = \max_i [t_i] + d$.

where t_i is the time delay of the circuitry in the ith stage of the pipeline.

d is the time delay of a latch, in general very much less than the $\max_i [t_i]$.

so in this problem $\max_i [t_i] = 45$ ns. $d = 5$ ns.

a. minimum cycle time is calculated as $t_p = 45 + 5 = 50$ ns.

b. Removing R_3 & R_4 the required time $t_n = 40 + 45 + 15$
$= 100$ ns.

c. speed up for 10 tasks $\rightarrow \frac{nt_n}{(k+n-1)t_p} = \frac{10 \cdot 100}{(3+10-1) \cdot 50}$
$= 1.67$.

speed up for 100 tasks $\Rightarrow \frac{nt_n}{(k+n-1)t_p} = \frac{100 \cdot 100}{(3+100-1) \cdot 50}$
$= 1.96$.

d. Maximum speed up possible $= t_n / t_p = 100/50 = 2$.

6. a. $t_p = \max_i [t_i] + d$
$= 95 + 5 = 100$ ns.

~~add 100 pairs no. of no.~~ no of tasks $n = 100$.
segment $k = 4$.

$\therefore$ Time to complete 100 tasks $= (n+k-1)t_p$
$= (100+4-1) \cdot 100$
$= 10300$ ns.
$= 10.3$ μs.

Tutorial 8:

1. $t_p = \max\{t_{p_i}\} + d$
   $= 90 + 10$
   $= 100$ ns.

   a. $(K+n-1) \cdot t_p$
   $= (4+1000-1) \cdot 100$
   $= 100300$ ns.

   b. $n \cdot t_n$
   $= 1000 \cdot (60+50+90+80)$
   $= 1000 \cdot 280$
   $= 280000$ ns.

   c. Throughput for pipelined execution
   = no. of instructions executed per unit time
   $= 1000 / 100300$ task/ns.
   $= 0.01$ task/ns.

2. $t_p = \max(150, 120, 160, 140) + 5$
   $= 165$ ns.
   $n = 1000 \quad K = 4 \quad \therefore$ total time $(1000+4-1) \cdot 165$
   $= 1003 \cdot 165$
   $= 165495$ ns
   $= 165.5$ μs

3. for four stage $t_{p_4} = \max(800, 500, 400, 300) + \text{delay}$
   $= 800 + 0 = 800$ picoseconds
   So execution time for 1 instruction = 1 clock cycle = 800 pico sec
   for two stage $t_{p_2} = \max(600, 350) + \text{delay}$
   $= 600 + 0 = 600$ picoseconds
   Execution time for 1 instruction = 600 picoseconds
   Throughput for 4 stage = 1 instruction / 800 ns.
   " " 2 stage = 1 instruction / 600 ns.

So increase $1/600 - 1/800$

So % of increase with respect to the previous one

$$\left(\frac{1/600 - 1/800}{1/800}\right) \times 100\ \%$$

$$= \left(\frac{200}{600 \times 800} \times 800\right) \times 100\ \%$$

$$= 33.33\ \%$$

4. cycle time in designing D1 = max(3,2,4,2,3) + delay

= 4 + 0 = 4 ns.

Execution time for 100 instructions in design D1.

= (K+n-1) · 4

= (5+100-1) · 4

= 416 ns.

Execution time for 100 instruction in design D2

= (K+n-1) · 2

= ~~208 ns~~

= (8+100-1) · 2

= 214 ns.

[cycle time = 2+0 → delay = 2ns.

So time saved = (416 - 214) = 202 ns.

5. P1 → cycle time = max (1,2,2,1) + delay = 2 ns

clock freq. = 1/2 = 0.5 GHz.

P2 → clock freq. = 1/max(1, 1.5, 1.5, 1.5) = 1/1.5 = 0.67 GHz

P3 → " = 1/max (0.5, 1, 1, 0.6, 1) = 1/1 = 1 GHz

P4 → " = 1/max (0.5, 0.5, 1, 1, 1.1) = 1/1.1 = 0.91 GHz.

so P3 has highest clock freq.

Tutorial 9:

1. RISC pipeline: Reduced instruction set computer (RISC)
   , An ability to use an efficient instruction pipeline.
   / ability to execute instructions at the rate of one per clock cycle.

say a three segment instruction pipeline has following phases:

I: Instruction
A: ALU operation
E: Execute Instruction.

Now, say the operation of following four instructions:

1. LOAD: $R1 \leftarrow M[address1]$
2. LOAD: $R2 \leftarrow M[address2]$
3. ADD: $R3 \leftarrow R1 + R2$
4. STORE: $M[address3] \leftarrow R_3$

Clock	1	2	3	4	5	6
1. LOAD R1	I	A	E			
2. LOAD R2		I	A	E		
3. ADD R_3			I	A	E	
4. STORE				I	A	E

clock	1	2	3	4	5	6	7
1. LOAD R1	I	A	E				
2. LOAD R2		I	A	E			
3. NO-operation			I	A	E		
4. ADD R3				I	A	E	
5. STORE					I	A	E

So here actually after LOAD R2 it is wasting a clock cycle. that is why it is called as pipeline timing with delayed load.

- Delayed Branch: say the following five/six instructions:

Load from memory to R1
increment R2
Add R3 to R4
Subtract R5 from R6
Branch to address X.
Next instruction in X.

On this delayed branch system no-operation instructions are being fetched from the memory and they are executed through the pipeline when the branch instruction is executed. It is upto the compiler to find useful instructions to put after the branch instruction. Failing that the compiler can insert no-op instructions.

Clock	1	2	3	4	5	6	7	8	9	10
1. Load	I	A	E							
2. increment		I	A	E						
3. Add			I	A	E					
4. Subtract				I	A	E				
5. Branch to X					I	A	E			
6. No-operation						I	A	E		
7. No-operation							I	A	E	
8. Next instruction in X.								I	A	E

2. LOAD $R_1 \leftarrow M[312]$
Add $R_2 \leftarrow R_2 + M[313]$
Increment $R_3 \leftarrow R_3 + 1$
STORE $M[314] \leftarrow R_3$

	1	2	3	4
	FI	DA	PO	EX.
		FI	DA	PO
			FI	DA
				FI

so seg EX: transfer memory word to R1
F0: Read M[313]
DA: Decode (increment) instruction
FI: Fetch the instruction from memory.

8. RISC → I LOAD R1 ← Memory[313]
A INCREMENT R1 ← R1+1
E

	Stage 1	2	3	4
instr 1.	I	A	E	
instr 2		I	A	E

Data hazards.

Before the completion of loading into R1 it is not possible to increment.

4. 4 floating-point pipeline processor.
each processor uses a cycle time of 40 ns.
total 400 floating-point operations are there.
so 400 operations will be divided into each of four processors
so processing time: $\frac{400}{4} \times 40 = 4000$ ns.

using a single pipeline, # cycle time is given 10 ns.
So processing time $400 \times 10 = 4000$ ns
so no change.

5. 250 billion floating-point operations so 250×10^9
100 megflop ⇒ 100 million floating point operation 100×10^6
so required time = $\frac{250 \times 10^9}{100 \times 10^6}$ sec.
= 2500 sec = 41.67 minutes.

Soln. Tutorial #10:

1. Binary equivalent of
12 → 1100
13 → 1101
14 → 1110
15 → 1111

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	1	1	0	0
0	0	0	0	1	1	0	1
0	0	0	0	1	1	1	0
0	0	0	0	1	1	1	1
← CS →						↓ RS_1	↓ RS_0

So, $CS = A_2 A_3 A_4' A_5' A_6' A_7'$
$RS_0 = A_0$
$RS_1 = A_1$
} for external circuit.

2.

Interface	Port A	Port B	Port C (control Reg.)	Port D (status Reg.)
no. # 1	10000000	10000001	10000010	10000011
2.	01000000	01000001	01000010	01000011
3.	00100000	00100001	00100010	00100011
4.	00010000	00010001	00010010	00010011
5.	00001000	00001001	00001010	00001011
6.	00000100	00000101	00000110	00000111

3.

inserted @ m bytes/sec. max. capcity K bytes.
deleted @ n bytes/sec.

a. empty buffer to fill when $m > n$, required time = $\frac{K}{m-n}$ sec
b. full buffer to empty when $m < n$, required time = $\frac{K}{n-m}$ sec
c. FIFO buffer is not required when $m = n$.

4. 1200 baud line → 1200 bits per second can be transmitted.

a. So for synchronous serial transmission no. of characters

$$= \frac{1200}{8}$$

$$= 150 \text{ char per second.}$$

b. Asynchronous serial transmission one bit for control signal.
with two stop bits.

so apart from 8 bits for a character 3 more bits

so total $8+3 = 11$ bits

so no. of characters $= \frac{1200}{11} = 109$ characters per second.

c. When asynchronous serial transmission ~~for~~ with one stop bit

so total $8+1+1 = 10$ bits.

so no. of characters $= \frac{1200}{10} = 120$ characters per second.

5. 20 MHz clock. i.e $\frac{1}{20} = 50$ ns clock pulse duration.
the access time to memory unit is 40 ns.