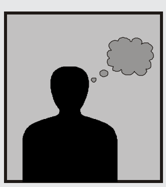
GATE

MADE EASY Workbook 2026



Detailed Explanations of Try Yourself *Questions*

Electronics Engineering

Computer Organization

MADE EASY PUBLICATIONS

1 Basics on CPU Registers and Memory

Detailed Explanation of Try Yourself Questions

T1 : Solution

(b)

T1 : Solution

(a)

T1 : Solution

(d)

■■■■

2 Machine Instructions and Addressing Modes

Detailed Explanation of Try Yourself Questions

T1 : Solution

(10)

As per the program and instruction length

Instruction 1	→	2 Words
Instruction 2	→	1 Word
Instruction 3	→	1 Word
Instruction 4	→	1 Word
Instruction 5	→	1 Word
Instruction 6	→	1 Word
Instruction 7	→	2 Words
Instruction 8	→	1 Word
		Total 10 Words

Irrespective of word length, if it is given as word addressable, then the entire memory location is taken as 1 word.

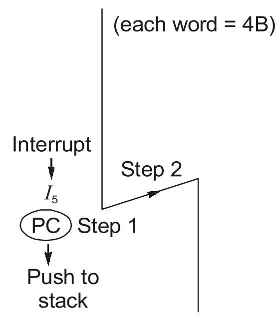
T2 : Solution

(c)

Given as Byte addressable, it means every byte of memory word has one address.

Program starts at memory location 1000, hence

	1000 : I_1	→ 2 Words
Loop :	1008 : I_2	→ 1 Word
	1012 : I_3	→ 1 Word
	1016 : I_4	→ 1 Word
	1020 : I_5 : INC R_3	→ 1 Word
	1024 : I_6	→ 1 Word
	1028 : I_7 : BNZ Loop	→ 2 Words
	1036 : I_8 : Halt	→ 1 Word
		Total 10 Words



When interrupt occurs at instruction I_5, the sequence of steps taken by CPU are

1. 'PC' content/next instruction address into stack memory i.e., address of $I_6 \Rightarrow 1024$.

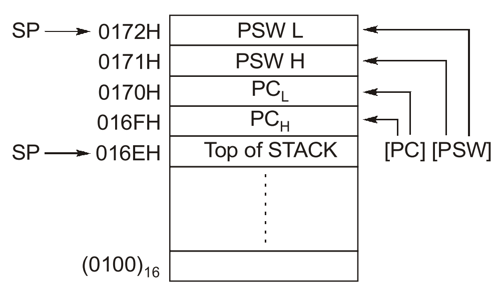
T3 : Solution

(d)

As per the sequence of events that take place when 'CALL' instruction is executed.

5FA0H : CALL Address Subroutine → 2 Words (So 4 bytes) 1 W = 2 Bytes

PC → 5FA4 H:



When CALL is executed.

Step 1 : [PC] → Stack Memory i.e., 2 B

∴ SP → Will be incremented by 2.

i.e., [SP] ← 0170H

Step 2: [PSW] → Stack Memory i.e., 2B

∴ SP → Will be again incremented by 2

∴ [SP] ← 0172H

T4 : Solution

(a)

T5 : Solution

(b)

f_{clk} = 1 GHz (given); Given, 4 cycles → Memory Reference, 2 cycles → Arithmetic computation

Addressing Mode		Number of Cycles
1. Register	→	0 (as per data)
2. Immediate	→	0 (as per data)
3. Direct	→	1 Memory cycle
4. Memory Indirect	→	2 Memory cycle + 1 Arithmetic computation
5. Indexed	→	1 Arithmetic computation + 1 Memory cycle
6. Auto indexed	→	1 Arithmetic computation + 1 Memory cycle

$$\text{Average CPI} = \sum_{i=1}^{m} \frac{IC_i}{IC} \times CPI_i \rightarrow \text{Cycles per } i^{th} \text{ instruction}$$

$\frac{IC_i}{IC}$ ⇒ Fraction of occurrence of i^{th} instruction

⇒ $[0.2 \times 0] + [0.2 \times 0] + 0.2[1 \times 4] + 0.1[2 \times 4] + 0.17[(1 \times 4) + (1 \times 2)] + 0.13[(1 \times 4) + (1 \times 2)]$

⇒ $0 + 0 + 0.8 + 0.8 + 1.02 + 0.78$

⇒ 3.4 cycles

Average Execution Time per instrution = $3.4 \times \frac{1}{1 \times 10^9}$

(or) here average fetch cycles for operands = 3.4 ns

For 1 instruction = 3.4 ns

$$\text{For 1 second} = \frac{1}{3.4ns}$$
$$= 0.294117 \times 10^9$$
$$= 294.11 \times 10^6$$
$$= 294.11 \text{ Million words/sec}$$

T6 : Solution

Given Instruction → ADD@300, 30(R3)

ADD@300 ↓ operand-1; 30(R3) ↓ operand-2

@ 300 → Indicates, Memory Indirect addressing mode

30(R3) → Index addressing mode

Operand-1; Effective address (E.A) = M[300]

Operand-2; EA = [R3] + 30

Memory Indirect addressing mode, Index addressing mode

T7 : Solution

(d)

$R0 \leftarrow \#35\text{ H};\quad R0\ \boxed{35H}$

$R1 \leftarrow \#FF\text{ H};\quad R1\ \boxed{FFH}$

$R1 \leftarrow [R0] \vee [R1]$

$[R1] = FFH$

35H ⟶ 0011 0101
FFH ⟶ 1111 1111
∨
1111 1111
FFH

T8 : Solution

(a)

■■■■

3 ALU, Data-path and Control Unit

Detailed Explanation of Try Yourself Questions

T1 : Solution

(c)

The machine here can support both horizontal and vertical μ-instructions.

Vertical → none/one of 3 μ-instructions

→ 2 bits are required for encoded form of 3 instructions.

Horizontal → 6 μ-operations → 6 bits

∴ Total → 2 + 6 → 8 bits

T2 : Solution

(c)

Vertical is most flexible, then horizontal type least is hardwired as no new instruction can be added and whose design should be done from the beginning.

T3 : Solution

(a, c)

As per data,

3 control signals → S_0, S_1 and S_3

3 Instructions → I_1, I_2 and I_3

3 μ-operations → T_1, T_2 and T_3

Expressions for S_0 and S_1 could be

$$S_0 = T_1[I_1 + I_2 + I_3] + T_2[0] + T_3[I_1]$$
$$= T_1 + T_3 I_1$$
$$S_1 = T_1[I_1 + I_3] + T_2[I_2] + T_3[I_2]$$

T4 : Solution

(b)

As pre data : Total instructions = 256

No. of μ-operations/instructions = 8

$$\text{Total memory locations} = 256 \times 8 = (2048)_{10}$$

$$\text{Address lines/bits} = 11 \text{ bits}$$

$$\text{No. of flags} = 16 \quad \therefore \quad \text{No. bits for coding flags} = 4$$

$$\text{No. of signals} = 48 \text{ i.e., 48 bits in } \mu\text{-instruction.}$$

$$\therefore \quad \text{Total no. of bits in one } \mu\text{-instruction} = (48 + 4 + 11) \text{ bits}$$

$$= 63 \text{ bits}$$

|← μ-instruction format for Horizontal C.U →|

Control bits 48 bits	Flags Condition	Address bits
	↓ 4 bits	↓ 11 bits

T5 : Solution

(10)

25 control signals → Given

Minimum code bits for 25 → 5 bits

$$2^n = 25$$

$$n = 5 \text{ at least}$$

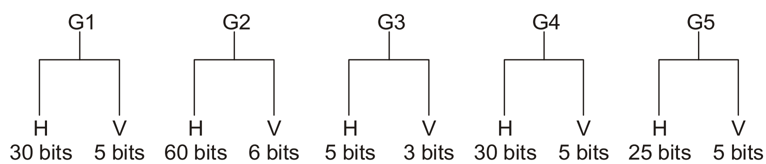
Control Signal-1	Control Signal-2
← 5 bits →	← 5 bits →

$$\Rightarrow \quad \text{Total} = 5 + 5 = 10 \text{ bits}$$

T6 : Solution

(126)

5 Groups of mutually exclusive signals i.e., both Horizontal and Vertical μ-programming is supported but one at a time.



Horizontal $\rightarrow$ Decoded form $\rightarrow 30 + 60 + 5 + 30 + 25 = 150$ bits

Vertical $\rightarrow$ Encoded form

$\rightarrow 5 + 6 + 3 + 5 + 5$

$\rightarrow$ 24 bits

Bits saved $\Rightarrow$ 150 – 24 bits

$\Rightarrow$ 126 bits

■■■■

4 Instruction Pipelining

Detailed Explanation of Try Yourself Questions

T1 : Solution

(b)

Design-1 : D1

$K = 5$ stages

$N = 100$

Non-uniform time delay stages

$\therefore \quad t_p$ = Max of (3ns, 2ns, 4ns, 2ns and 3ns)

$= 4$ ns

Total time 1 $= (K + N - 1)t_p$

$= (5 + 100 - 1)$ 4ns

$= 416$ ns

Design-2 : D2

$K = 8$

$N = 100$

$t_p = 2$ ns (given)

Total time 2

$= (K + N - 1)t_p$

$= (8 + 100 - 1)$2ns

$= 107 \times 2$ns

$= 214$ ns

Time saved $= 416$ ns $- 214$ ns $= 202$ ns

T2 : Solution

(13)

$K = 4$ stages, assume them as Fetch, Decode, Execute and Memory Access, i.e., F, Dec, Ex, M.A.

Based on the given data the overlapped execution would be

Clock	1	2	3	4	5	6	7	8	9	10	11	12	(13)	14	15	16
I_1	F	D	D	Ex	M.A	M.A										
I_2		F	F	D	Ex	Ex	M.A									
I_3				F	D		Ex	Ex	M.A							
I_4					F	F	D		Ex	Ex	M.A					
I_1							F		D	D	Ex	M.A	M.A			

I_1 Completes Execution

I_1 completes execution in $i = 2$ @ 13 clock.

T3 : Solution

(c)

T4 : Solution

(c)

T5 : Solution

(33.28)

T6 : Solution

(b)

T7 : Solution

(1.4)

■■■■