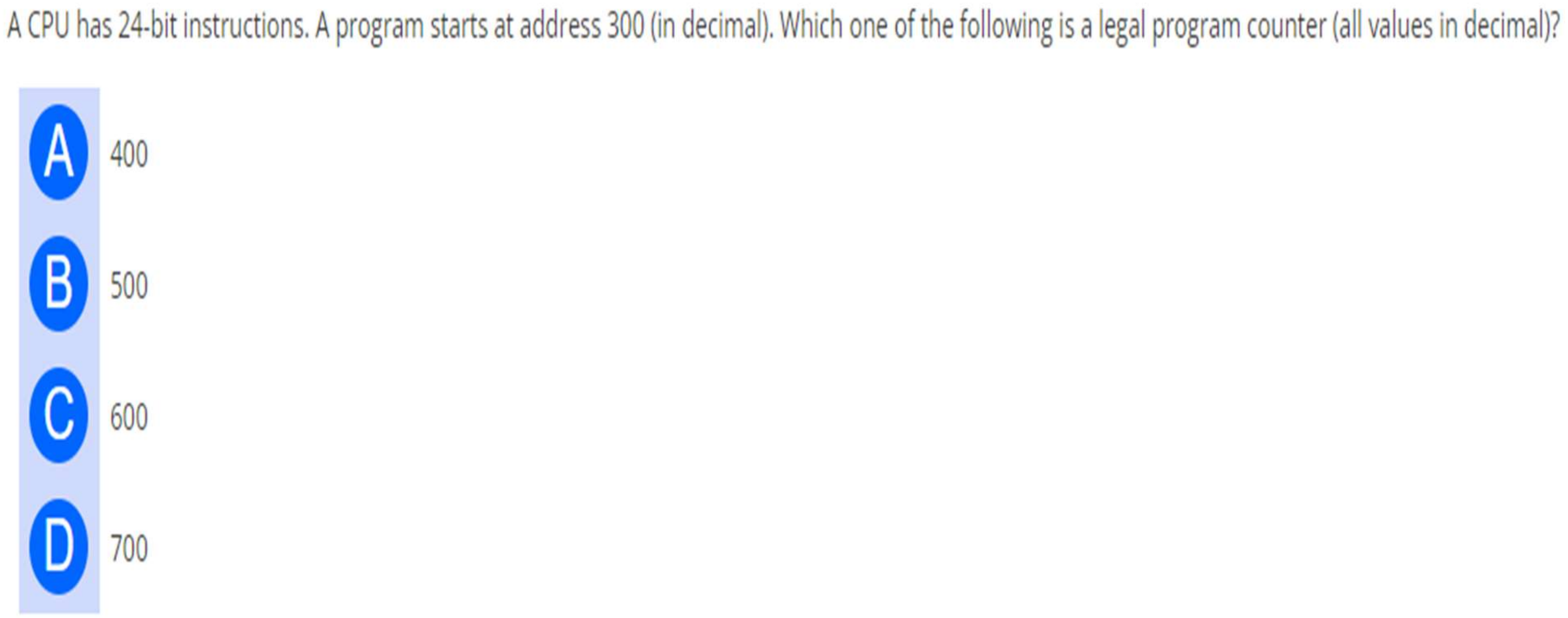
Tutorial -3

Q1



A CPU has 24-bit instructions. A program starts at address 300 (in decimal). Which one of the following is a legal program counter (all values in decimal)?

- A 400
- B 500
- C 600
- D 700

Default : Byte Addressable

Hence

PC +1 Points to the address of the next instruction itself
Hence PC + 1 = PC + WordSize/Byte

Every Instruction is of 3 Bytes (24 Bits)
SO PC will be in the multiple of 3 (PC = PC + 24/3)

Valid PC value would be 300,303,306………

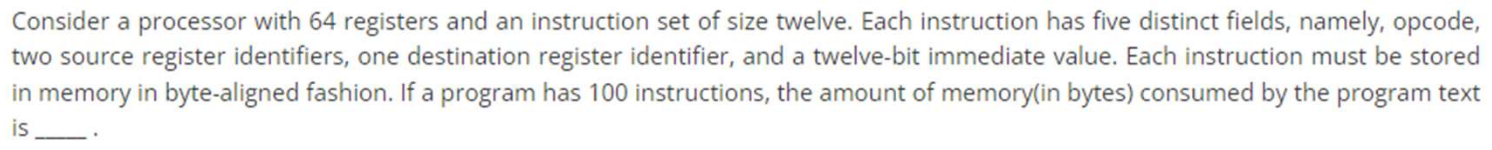
600 is the correct option

Q2

A processor has 40 distinct instructions and 24 general purpose registers. A 32-bit instruction word has an opcode, two register operands and an immediate operand. The number of bits available for the immediate operand field is ______ .

Opcode ceil(log 40) = 6	Register ceil(log 24) = 5	Register ceil((log 24) =5	32 - 16 = 16 bits for Immediate

Q3



Consider a processor with 64 registers and an instruction set of size twelve. Each instruction has five distinct fields, namely, opcode, two source register identifiers, one destination register identifier, and a twelve-bit immediate value. Each instruction must be stored in memory in byte-aligned fashion. If a program has 100 instructions, the amount of memory(in bytes) consumed by the program text is _____ .

Answer: 500 bytes

Number of registers $= 64$

Number of bits to address register $= \lceil \log_2 64 \rceil = 6 - \text{bits}$

Number of Instructions $= 12$

Opcode size $= \lceil \log_2 12 \rceil = 4$

Opcode(4)	reg1(6)	reg2(6)	reg3(6)	Immediate(12)

Total bits per instruction $= 34$

Total bytes per instruction $= 4.25$

Due to byte alignment we cannot store 4.25 bytes, without wasting 0.75 bytes.

So, total bytes per instruction $= 5$

Total number of instructions $= 100$

Total size $=$ Number of instructions $\times$ Size of an instruction

$$= 100 \times 5 = 500 \text{ bytes}$$

In your own words, describe the Fetch Decode Execute Cycle

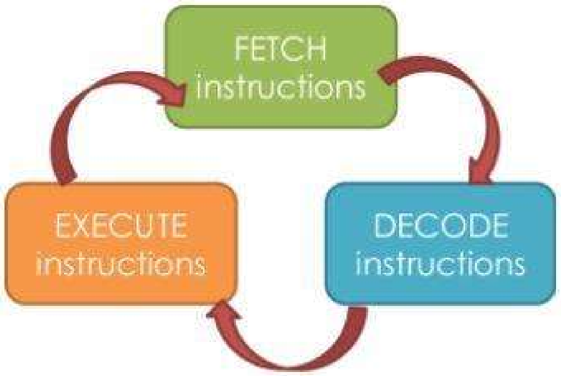
1) In the fetch step, the CPU sends the address of the instruction it wants to the RAM using a hardware path known as the 'address bus'. The RAM then finds the instruction stored at the address and sends it back to the CPU using the 'data bus'. The instruction is then stored in something known as a register which is the CPU's own memory space.

The Fetch-Decode-Execute Cycle

A standard process describes the steps needed for processing to take place.

It is called the **Fetch - Decode - Execute cycle** or sometimes simply called the Fetch-Execute Cycle.

First of all, both the data and the program that acts upon that data are loaded into main memory (RAM) by the operating system. The CPU is now ready to do some work.



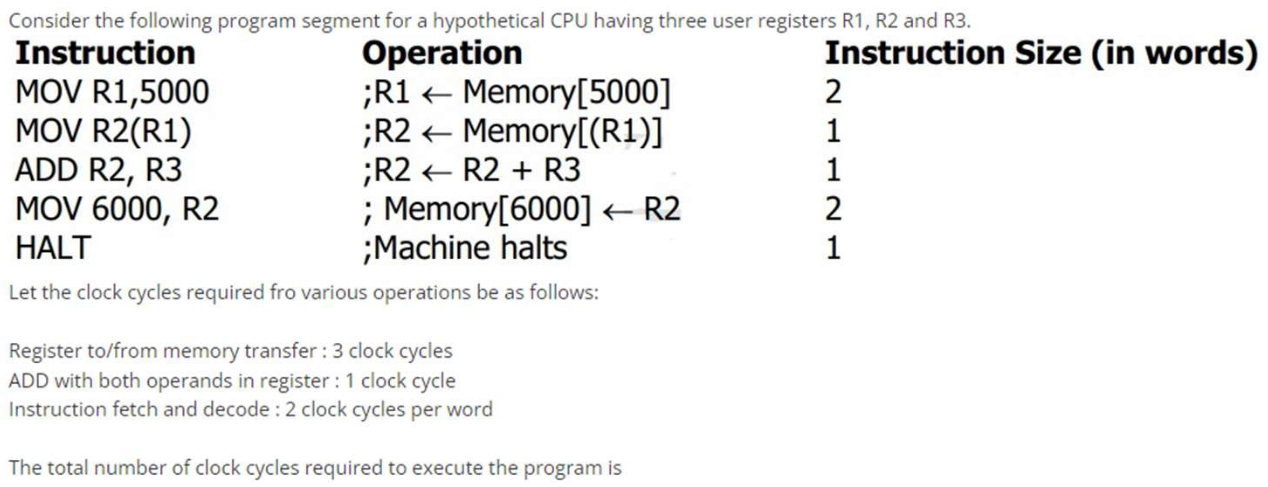
2) The decode stage is basically just where the CPU breaks down the instruction it received into something it can easily understand.

The CPU identifies the operation code. If this code needs the CPU to act on some data, then the second half of the operation code will either have the data it needs or a memory address of where it's being stored.

3) The execute step processes the instruction provided. It carries out the instruction in the operation code using the fetched data. The results of this are stored in a different register, ready to either be used in more instructions or to be sent all the way back to the main memory.

The CPU then repeats this cycle.

Q4



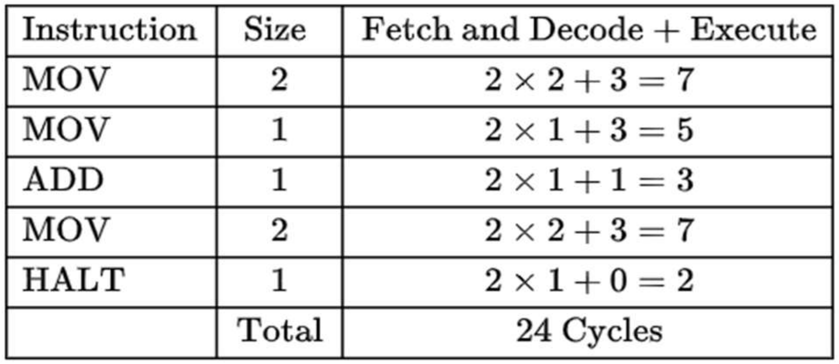
Consider the following program segment for a hypothetical CPU having three user registers R1, R2 and R3.

Instruction	Operation	Instruction Size (in words)
MOV R1,5000	;R1 ← Memory[5000]	2
MOV R2(R1)	;R2 ← Memory[(R1)]	1
ADD R2, R3	;R2 ← R2 + R3	1
MOV 6000, R2	; Memory[6000] ← R2	2
HALT	;Machine halts	1

Let the clock cycles required fro various operations be as follows:

Register to/from memory transfer : 3 clock cycles
ADD with both operands in register : 1 clock cycle
Instruction fetch and decode : 2 clock cycles per word

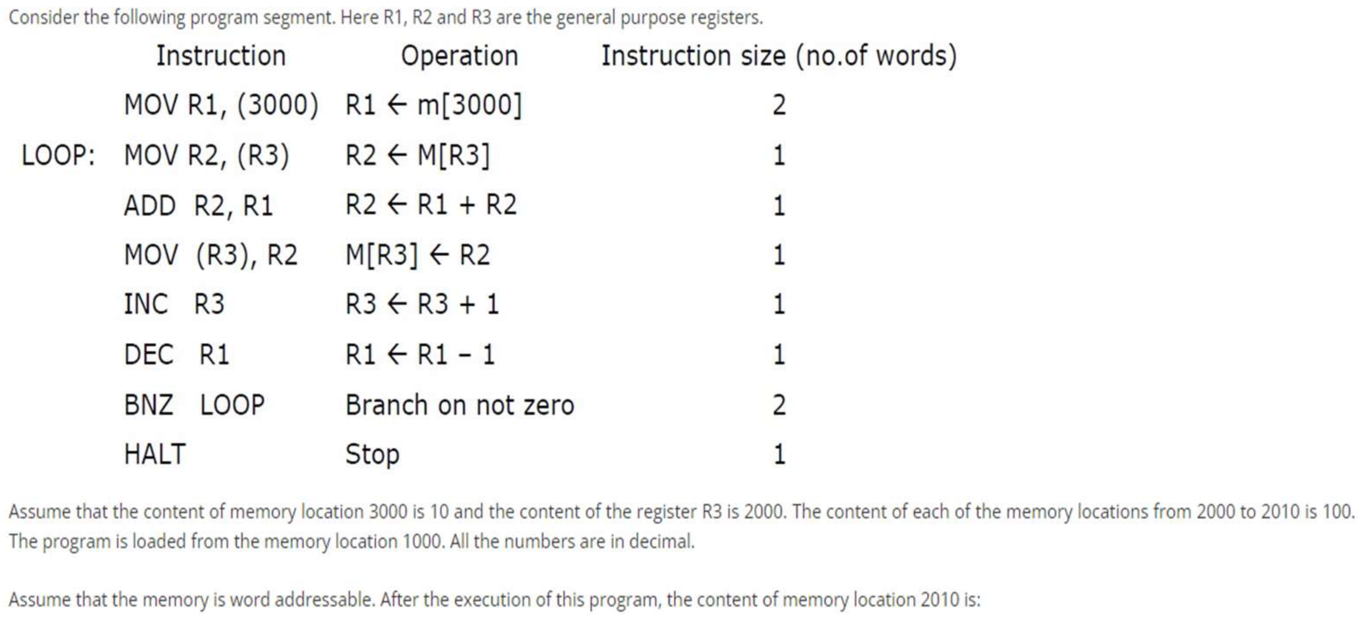
The total number of clock cycles required to execute the program is



Instruction	Size	Fetch and Decode + Execute
MOV	2	$2 \times 2 + 3 = 7$
MOV	1	$2 \times 1 + 3 = 5$
ADD	1	$2 \times 1 + 1 = 3$
MOV	2	$2 \times 2 + 3 = 7$
HALT	1	$2 \times 1 + 0 = 2$
	Total	24 Cycles

12 Cycles Execution for Fetch Decode
HALT Will take NO Clock cycles to execute

Q5

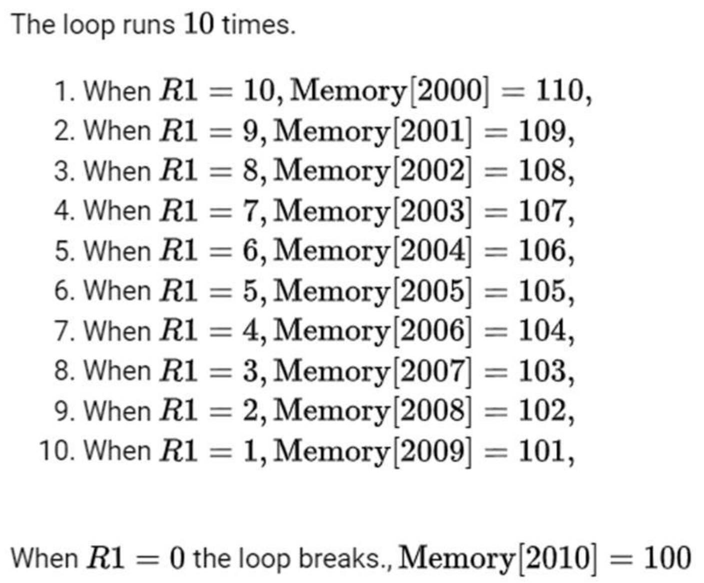


Consider the following program segment. Here R1, R2 and R3 are the general purpose registers.

	Instruction	Operation	Instruction size (no.of words)
	MOV R1, (3000)	R1 ← m[3000]	2
LOOP:	MOV R2, (R3)	R2 ← M[R3]	1
	ADD R2, R1	R2 ← R1 + R2	1
	MOV (R3), R2	M[R3] ← R2	1
	INC R3	R3 ← R3 + 1	1
	DEC R1	R1 ← R1 – 1	1
	BNZ LOOP	Branch on not zero	2
	HALT	Stop	1

Assume that the content of memory location 3000 is 10 and the content of the register R3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the numbers are in decimal.

Assume that the memory is word addressable. After the execution of this program, the content of memory location 2010 is:


The loop runs 10 times.

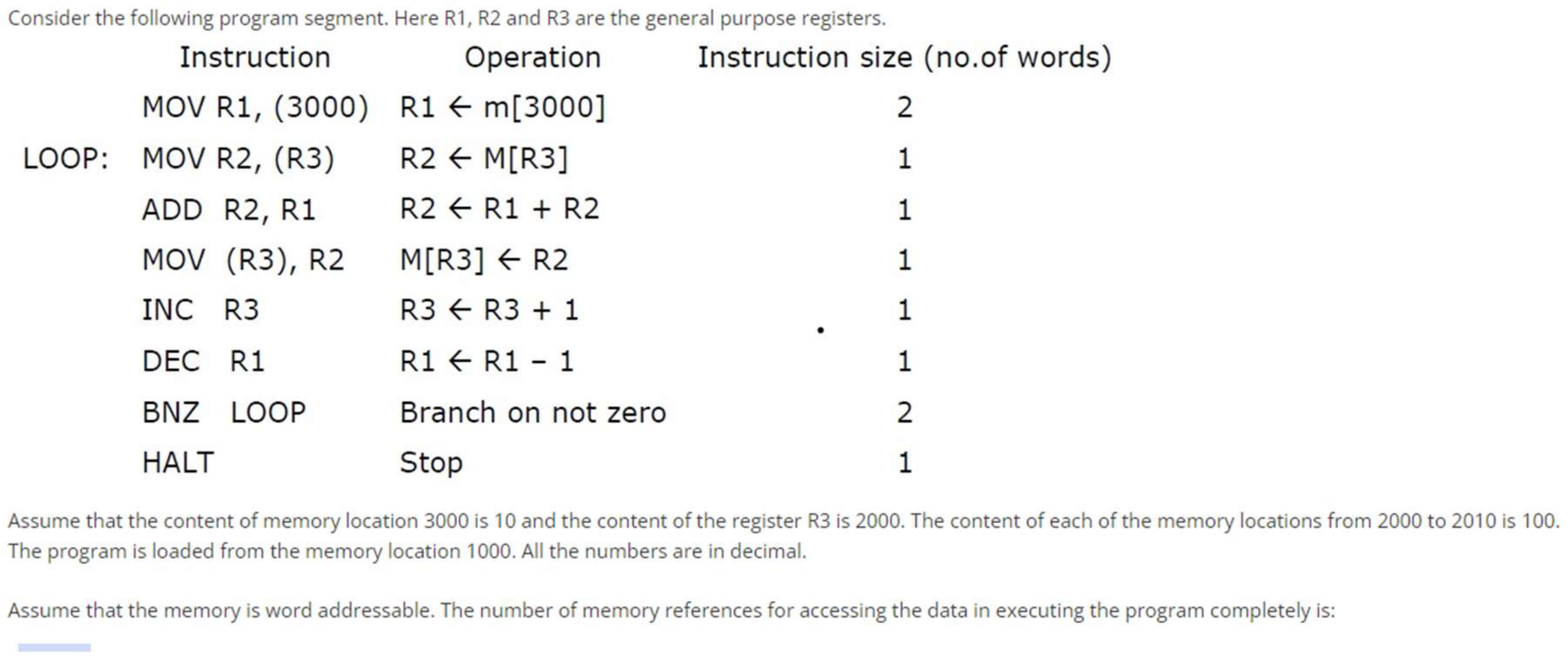
1. When $R1 = 10, \mathrm{Memory}[2000] = 110,$
2. When $R1 = 9, \mathrm{Memory}[2001] = 109,$
3. When $R1 = 8, \mathrm{Memory}[2002] = 108,$
4. When $R1 = 7, \mathrm{Memory}[2003] = 107,$
5. When $R1 = 6, \mathrm{Memory}[2004] = 106,$
6. When $R1 = 5, \mathrm{Memory}[2005] = 105,$
7. When $R1 = 4, \mathrm{Memory}[2006] = 104,$
8. When $R1 = 3, \mathrm{Memory}[2007] = 103,$
9. When $R1 = 2, \mathrm{Memory}[2008] = 102,$
10. When $R1 = 1, \mathrm{Memory}[2009] = 101,$

When $R1 = 0$ the loop breaks., $\mathrm{Memory}[2010] = 100$

Execution Trace of first loop

1st Instruction -> R1=10
2nd Instruction -> R2= 100
3rd Instruction -> R2=110
4th Instruction -> 2000 Memory location has value 110
5th Instruction -> R3=2001
6th Instruction -> R1 = 9
7th Instruction R1 Not Zero, FLAG NOT SET Hence LOOP Back to LABEL LOOP

Q6



Consider the following program segment. Here R1, R2 and R3 are the general purpose registers.

	Instruction	Operation	Instruction size (no.of words)
	MOV R1, (3000)	R1 ← m[3000]	2
LOOP:	MOV R2, (R3)	R2 ← M[R3]	1
	ADD R2, R1	R2 ← R1 + R2	1
	MOV (R3), R2	M[R3] ← R2	1
	INC R3	R3 ← R3 + 1	1
	DEC R1	R1 ← R1 – 1	1
	BNZ LOOP	Branch on not zero	2
	HALT	Stop	1

Assume that the content of memory location 3000 is 10 and the content of the register R3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the numbers are in decimal.

Assume that the memory is word addressable. The number of memory references for accessing the data in executing the program completely is:

We only care about the Data access here as part of Execute

Inside Loop:

MOV R2 [R3]
MOV [R3] R2

Word Addressable Hence total no of memory references = 2*10 = 20

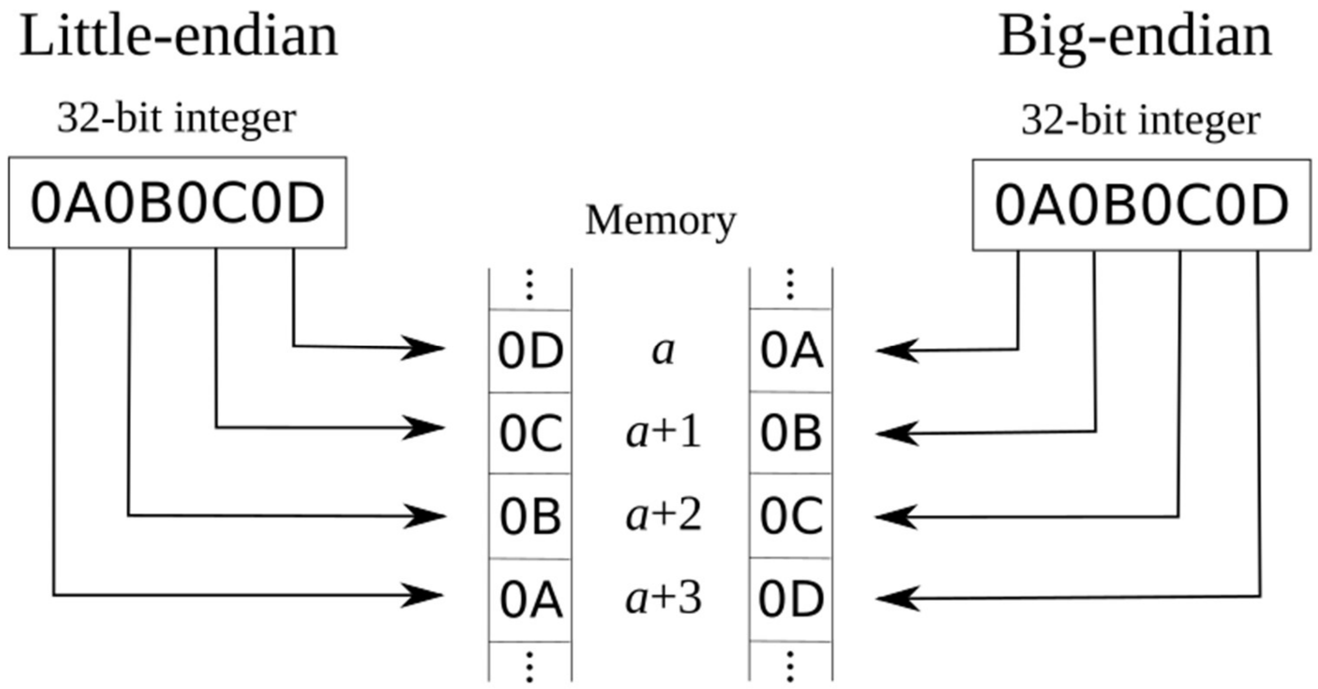
Outside Loop:

Size of instruction is 2 word bt size of memory operand is of 1 word
if u r fetching instruction= 2 mem ref
fetching operand= 1 mem ref

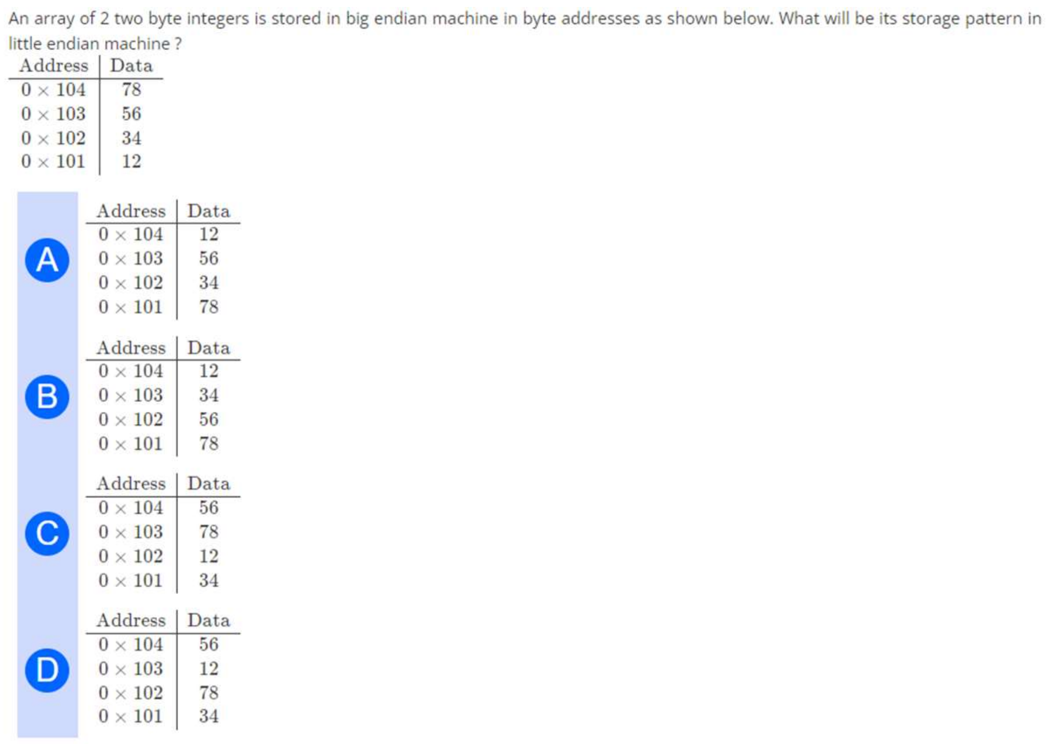
Hence Answer = 20+1 = 21 Clock Cycles

Accessing the data both include reading and storing data to main memory

Endianess


Little-endian
32-bit integer
0A0B0C0D
Memory
0D
0C
0B
0A
a
a+1
a+2
a+3
0A
0B
0C
0D
Big-endian
32-bit integer
0A0B0C0D

Q7 Take home (Answer on next Slide)



An array of 2 two byte integers is stored in big endian machine in byte addresses as shown below. What will be its storage pattern in little endian machine ?

Address	Data
0×104	78
0×103	56
0×102	34
0×101	12

A

Address	Data
0×104	12
0×103	56
0×102	34
0×101	78

B

Address	Data
0×104	12
0×103	34
0×102	56
0×101	78

C

Address	Data
0×104	56
0×103	78
0×102	12
0×101	34

D

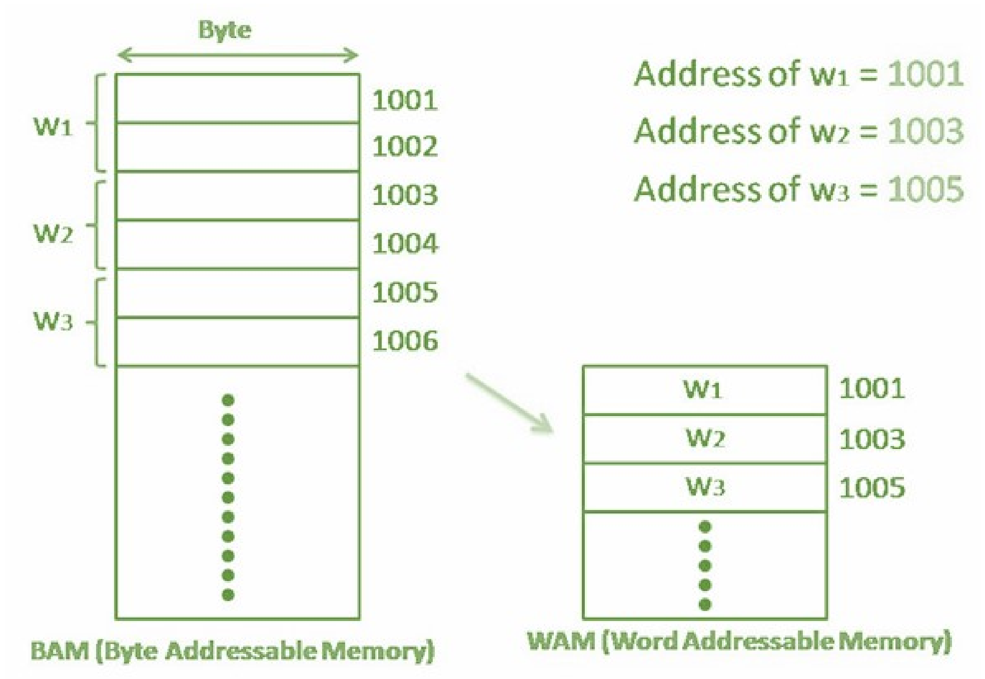
Address	Data
0×104	56
0×103	12
0×102	78
0×101	34

C.

Address	Data
0×104	56
0×103	78
0×102	12
0×101	34

Word Addressable and Byte Addressable

- The Processor Word Size of Processor is the amount of bits it can process at one go. If it’s a 64 Bit microprocessor then a Word size for the Processor is 64
- Word Size of the Processor is separate from addressability of the memory. Memory can be byte addressable or Word Addressable.
- If a Memory is word addressable as shown in write then each block of the memory would be equal to word Length
- Memory are usually Byte Addressable and hence you may need to fetch multiple blocks of bytes which add up to word size for a processor



Word Addressable and Byte Addressable Access

	No of words	Word Addressability Mem reference	Byte Addressability Mem Reference
16 Bit Microprocessor	1	1	2
32 Bit Microprocessor	1	1	4
32 Bit Microprocessor	2	2	8