CS & IT ENGINEERING

COMPUTER ORGANIZATION AND ARCHITECTURE

CPU & Control Unit

Lecture No.-04

By- Vishvadeep Gothi sir

Recap of Previous Lecture

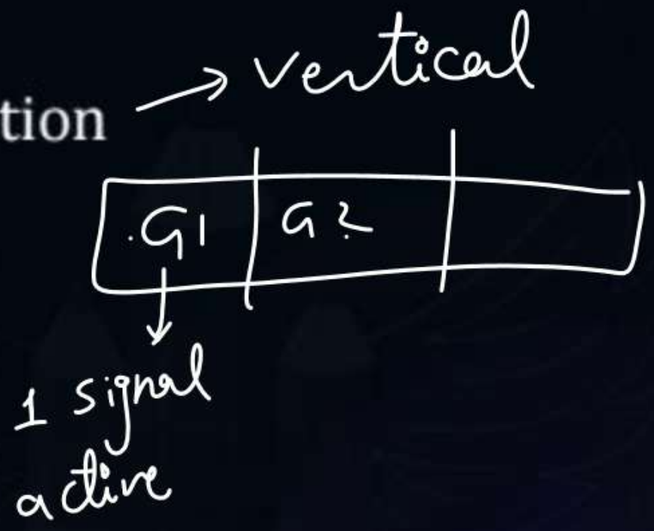
Topic CPU

Topic MIPS

Topic Data Path

Topics to be Covered

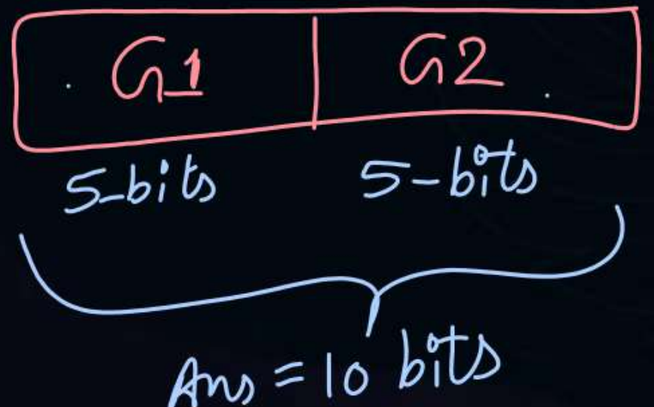
- **Topic** Datapath
- **Topic** Control Unit Organization
- **Topic** Byte Ordering



[NAT]

#Q. A micro-programmed control unit is required to generate a total of 25 control signals. Assume that during any microinstruction at most 2 control signals are active. Minimum number of bits required in the control word to generate the required control signals will be?

among 25 signal → max 2 signal at a time active

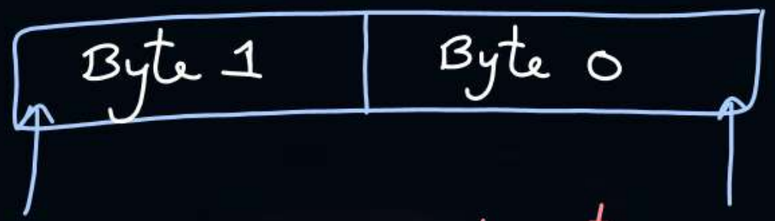


assume
25 signals ⇒ S1, ···, S25

Topic : Byte Ordering

Assume

Data ⇒ 2 B



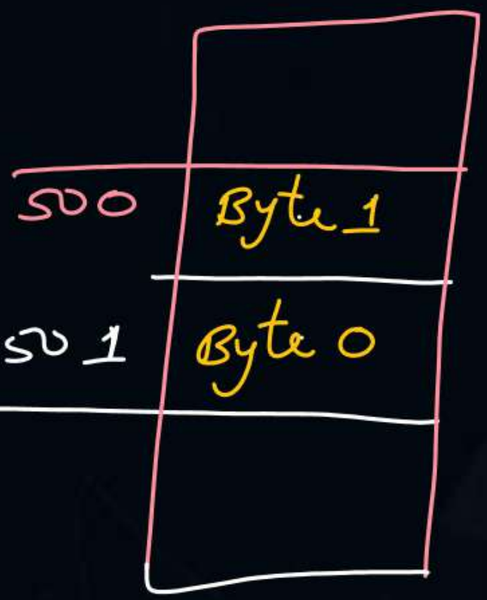
Most significant byte

Least significant byte

Big endian

store in (byte addressable) mem.

starting add. ⇒ 500

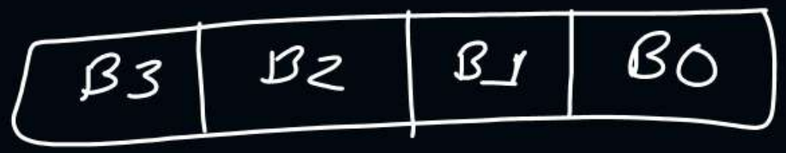


Little endian

Topic : Byte Ordering

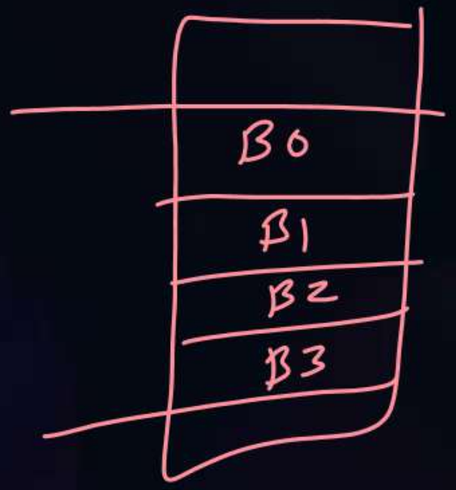
ex:-

4B data



big endian →

↓ little endian



[NAT]

#Q. Consider a 6-words instruction, which is of the following type:

1 word = 4 bytes

destinatn

Opcode	Mode1	Mode2	Address1	Address2

Reg. ind. mode indirect

The first operand (destination) uses register indirect mode and second operand uses indirect mode. Assume each operand is of size 2 words, each address is of 2 words and main memory takes 50ns for 1 byte access. Further assume that the opcode denotes addition operation which copies result of addition of 2 operands. Total time required in:

1. Fetch cycle of instruction 1200 nsec
2. Execution cycle of instruction 1600 nsec
3. Instruction cycle of instruction 1200 + 1600 = 2800ns

Inst^{n} fetch ⇒

6 word mem. read time = 6 * 200
= 1200 ns

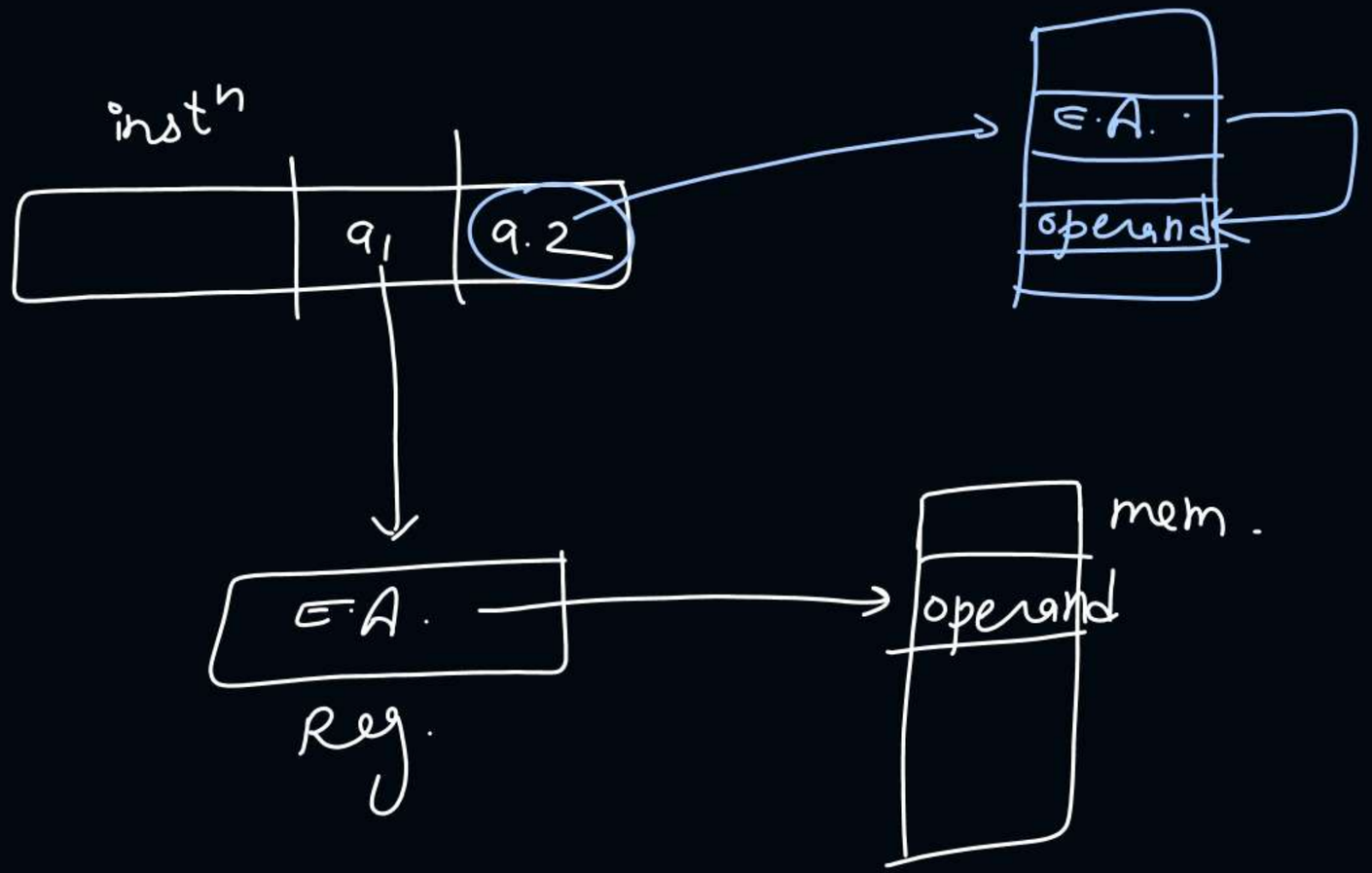
for 1B, mem. = 50 nsec

for 1 word ⇒ 4 * 50
= 200 ns

Execution cycle ⇒

	operand 1	operand 2
E.A. calculat^{n} & operand fetch	8 * 50 = 400 nsec	(8+8) * 50 = 800 ns
write back result	400 nsec	

Total = 400 + 800 + 400
= 1600 nsec


inst^n
q1
q2
E.A.
operand
E.A.
Reg.
mem.
operand

jump or branch

branch type instn

unconditional branch

⇓

always branch taken

⇓

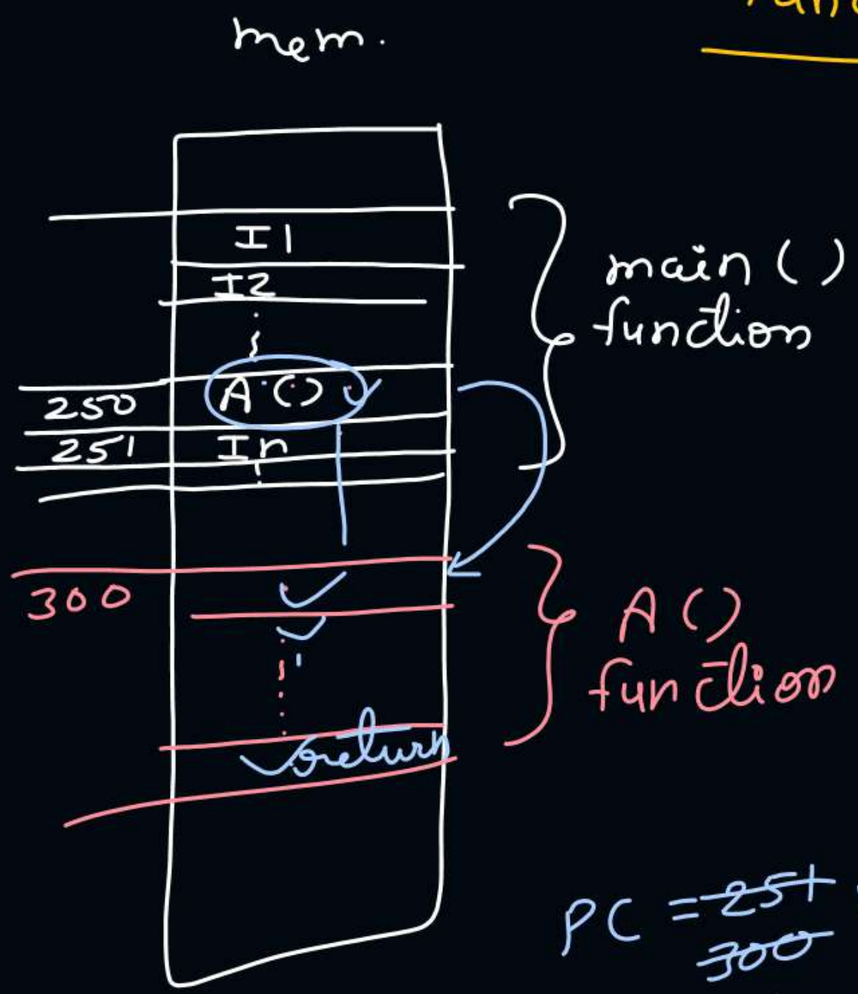
conditional

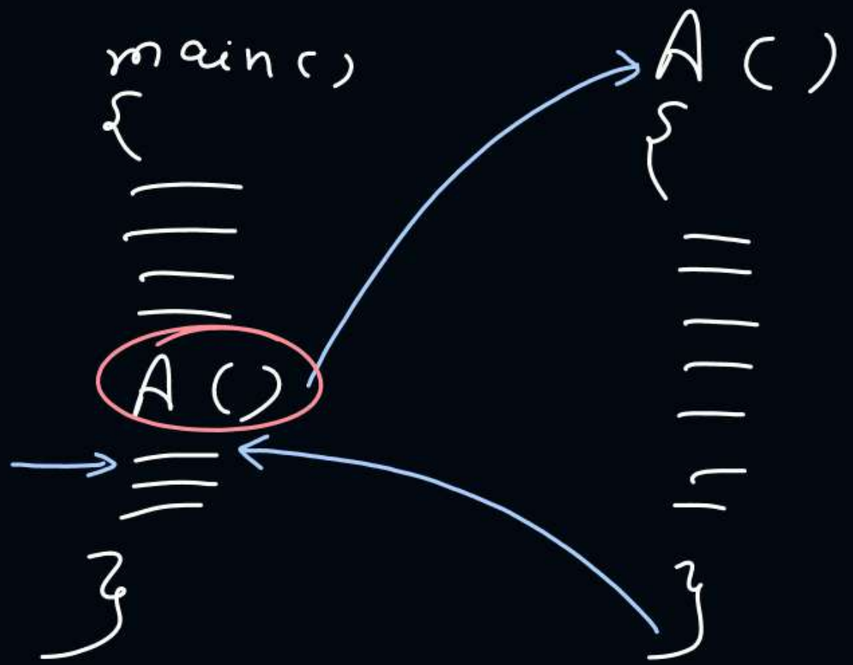
- if condit^{n} true ⇓ Branch taken
- if false ⇓ Not taken

function call (unconditional jump)

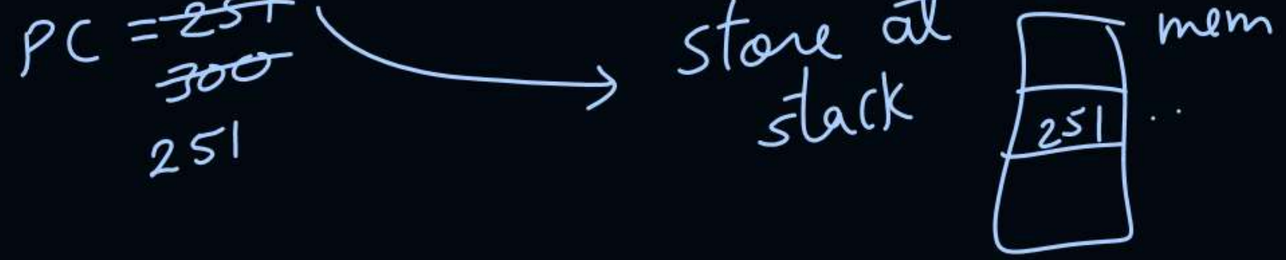
1. store current PC value on stack
2. Jump to called funct^{n} by updating PC by it's starting add.

Function call





PC = ~~251~~ ~~300~~ 251



[NAT]

Quest^{n} on next page

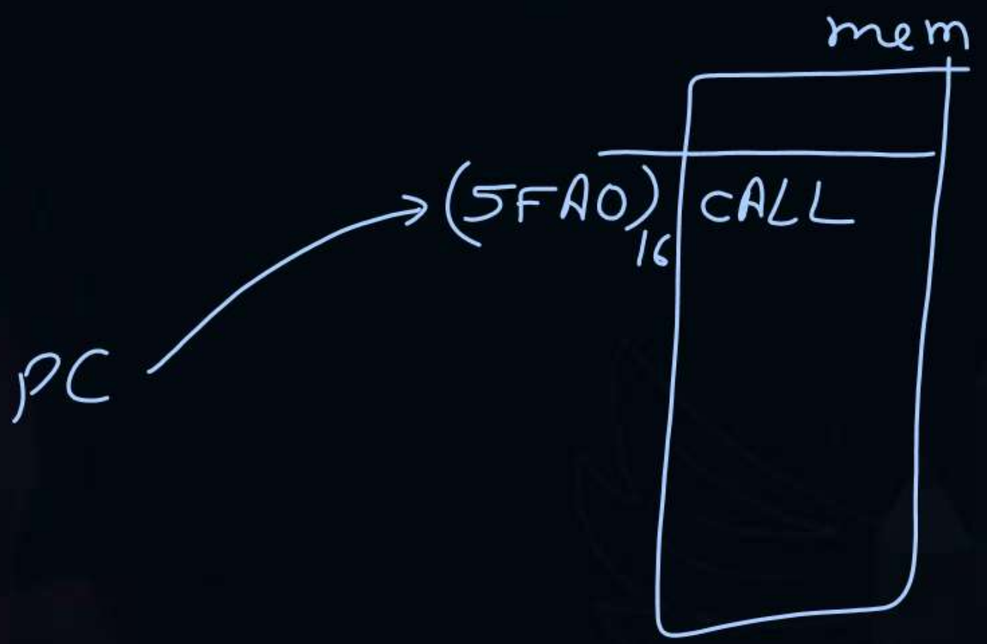
The content of PC just before the fetch of a CALL instruction is $(5FA0)_{16}$. After execution of the CALL instruction, the value of the stack pointer is

A. $(016A)_{16}$

B. $(016C)_{16}$

C. $(0170)_{16}$

D. $(0172)_{16}$



[NAT] GATE-15 (set-2)

#Q. Consider a processor with byte-addressable memory. Assume that all registers, including Program Counter (PC) and Program Status Word (PSW), are of size 2 bytes. A stack in the main memory is implemented from memory location $(0100)_{16}$ and it grows upward. The stack pointer (SP) points to the top element of the stack. The current value of SP is $(016E)_{16}$. The CALL instruction is of two words, the first word is the op-code and the second word is the starting address of the subroutine (one word = 2 bytes). The CALL instruction is implemented as follows:

- Store the current value of PC in the stack.
- Store the value of PSW register in the stack.
- Load the starting address of the subroutine in PC.

Byte add. mem.

all Reg. (PC, PSW) ⇒ 2 Bytes

CALL $inst^{n}$ ⇒ 2 words

1 word	1 word
opcode	starting add. of subroutine

← 4 bytes →

SP

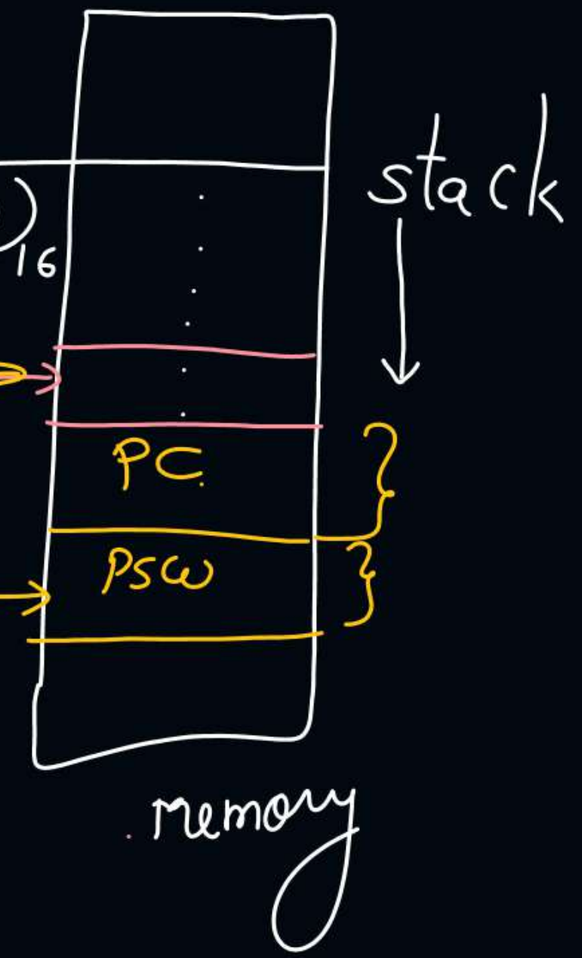
$(016E)_{16}$

+2

0170

+2

0172



	PC
Before fetch of CALL	$(5FA0)_{16}$
After —11—	$(5FA4)_{16}$
After Execution phase	starting add. of subroutine

2 mins Summary

Topic Byte Ordering

Topic Function Call

THANK - YOU