COMPUTER SCIENCE

Computer Organization and Architecture

Machine Instruction and Addressing Modes

Lecture_04

Vijay Agarwal sir

01 Expand Opcode Technique

02 Addressing Modes

Instruction Format.

1. Stack Based org.
2. Accumulator Based org.
3. General Register org.

① STACK Based org: [Stack - CPU]

↳ TOS [Top of the Stack].

eg1 $A+B$

eg2 $(x*y)+z$.

eg3 $X=(A+B)*(C+D)$

] STACK-CPU.

② Accumulator Based org.

ALU operation Destn S1 S2

AC AC Reg/mem

eg1 (A + B)

eg2 (x * y) + z

eg3 X = [A+B] * [C+D]

Instruction Set Architecture

CUP Organization is classified into 3 types based on the availability of ALU Operand (Data) (AF: Address field or AI: Address Instruction)

1. Stack-CPU **[0AF]**
2. Accumulator-CPU **[1AF]**
3. General Register organization
   i. Reg-Memory reference CPU **[2AF]**
   ii. Reg-Reg reference CPU **[3AF]**

Stack Organization

X = (A + B) × (C + D)

I_1:	PUSH	A	TOS ← A
I_2:	PUSH	B	TOS ← B
I_3:	ADD		TOS ← (A + B)
I_4:	PUSH	C	TOS ← C
I_5:	PUSH	D	TOS ← D
I_6:	ADD		TOS ← (C + D)
I_7:	MUL		TOS ← (C + D) × (A + B)
I_8:	POP	X	M[X] ← TOS

8 Machine Instruction Required (Stack-CPU)

Single Accumulator Organization

$$X = (A + B) \times (C + D)$$

I_1:	LOAD	A	AC ← M[A]
I_2:	ADD	B	AC ← AC + M[B]
I_3:	STORE	T	M[T] ← AC
I_4:	LOAD	C	AC ← M[C]
I_5:	ADD	D	AC ← AC + M[D]
I_6:	MUL	T	AC ← AC × M[T]
I_7:	STORE	X	M[x] ← AC

7 Machine Instruction Required (AC-CPU)

General Register Organization

Based on the Number of Registers (Register bike) Supported by the Processor this Arch. is divided into 2 Types.

1. Register – mem Ref CPU.
2. Register – Register Ref CPU.

General Register Organization

① Register – Memory Reference : This Architecture support less Number of Registers so Register file size is small.

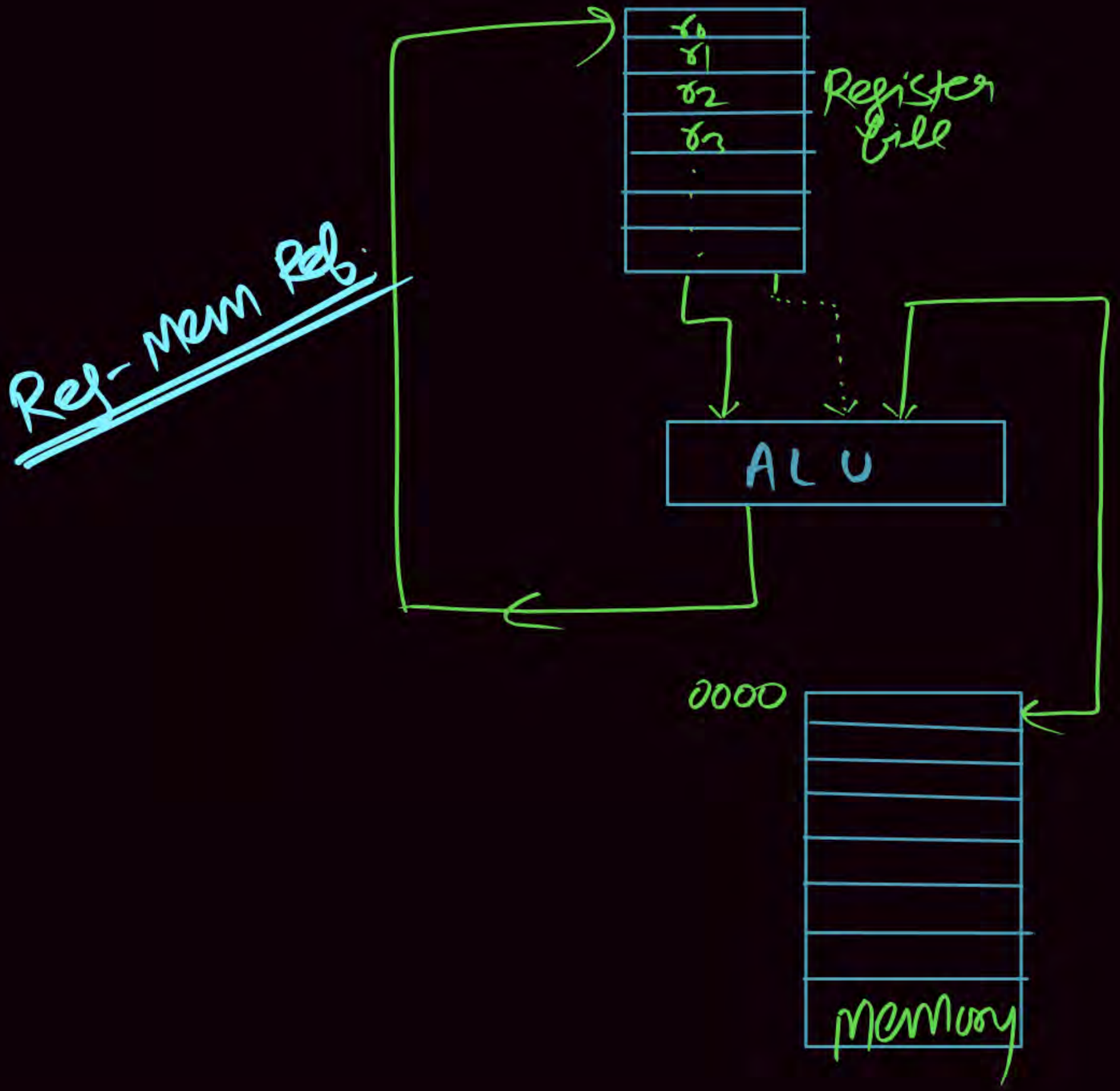
Compitable Format ⇓ ALU operation

	Destn	Source 1	Source 2
	Reg(S1)	Reg	Reg/mem

DATA Transfer Operation

Destn	Source	
Reg	Mem	'MOV'
Mem	Reg	
Reg	Reg	

∴


R0
R1
R2
R3
Register File
Reg-Mem Reg
ALU
0000
memory

eg ADD R_1 R_2

$R_1 \leftarrow R_1 + R_2$

Destn (S_1) — S_1 (Reg) — S_2 (Reg)

eg2 ADD R_1 [6000]

$R_1 \leftarrow R_1 + M[6000]$

Destn (Reg) — S_1 (Reg) — S_2 (Mem)

eg MOV γ_1 γ_2

$\gamma_1 \leftarrow \gamma_2$

Destn (Reg) — Source (Reg)

eg MOV γ_5 [1000]

$\gamma_5 \leftarrow M[1000]$

Destn (Reg.) — S (Memory)

eg MOV [6000] γ_2

$M[6000] \leftarrow \gamma_2$

Destn (Memory) — Source (Reg.)

(eg) (A + B) Using Reg - Mem Ref. How many m/c Instn Required ?

I_1 : MOV R_0 A; $R_0 \in M[A]$

I_2 ADD R_0 B; $R_0 \in \underbrace{R_0}_{S_1\ (Reg)} + \underbrace{M[B]}_{S_2\ (Mem)}$

---

2 Machine Instn (Using Reg - Mem Ref.)

General Register Organization

$(x*y)+z$; Using Reg-Mem Ref CPU ?

I_1: MOV r_0 X; $r_0 \leftarrow M[X]$

I_2: MUL r_0 y; $r_0 \leftarrow r_0 * M[y]$ $r_0 = x*y$

I_3: ADD r_0 z; $r_0 \leftarrow r_0 + M[z]$ $r_0 = (x*y)+z$.

3 m/c Instn using Reg-Mem Ref.

General Register Organization

$X = (A+B) * (C+D)$

A, B, C, D, X are variables in memory, then How many M/C Instn Required Using Reg-Mem Ref?

I_1: MOV R_0 A; $\quad R_0 = A$ $\quad R_0 \in R_0 + M[B]$

I_2: ADD R_0 B; $\quad R_0 = A+B$

I_3: MOV R_1 C; $\quad R_1 = C$ $\quad R_1 \in R_1 + M[D]$

I_4: ADD R_1 D; $\quad R_1 = C+D$

I_5: MUL R_0 R_1; $\quad R_0 = R_0 * R_1$ $\quad R_0 = (A+B) * (C+D)$

I_6: MOV X R_0; $\quad M[X] = (A+B) * (C+D)$

6 M/c Instn Using Reg-mem.

General Register Organization

$$X = (A + B) \times (C + D)$$

[Reg - mem Reference].

I_1:	MOV	R1, A	R1 ← M[A]
I_2:	ADD	R1, B	R1 ← R1 + M[B]
I_3:	MOV	R2, C	R2← M[C]
I_4:	ADD	R2, D	R2 ← R2 + M[D]
I_5:	MUL	R1, R2	R1 ← R1 × R2
I_6:	MOV	X, R1	M[X] ← R1

6 Machine Instruction Required (Reg-CPU)

General Register Organization [RISC]

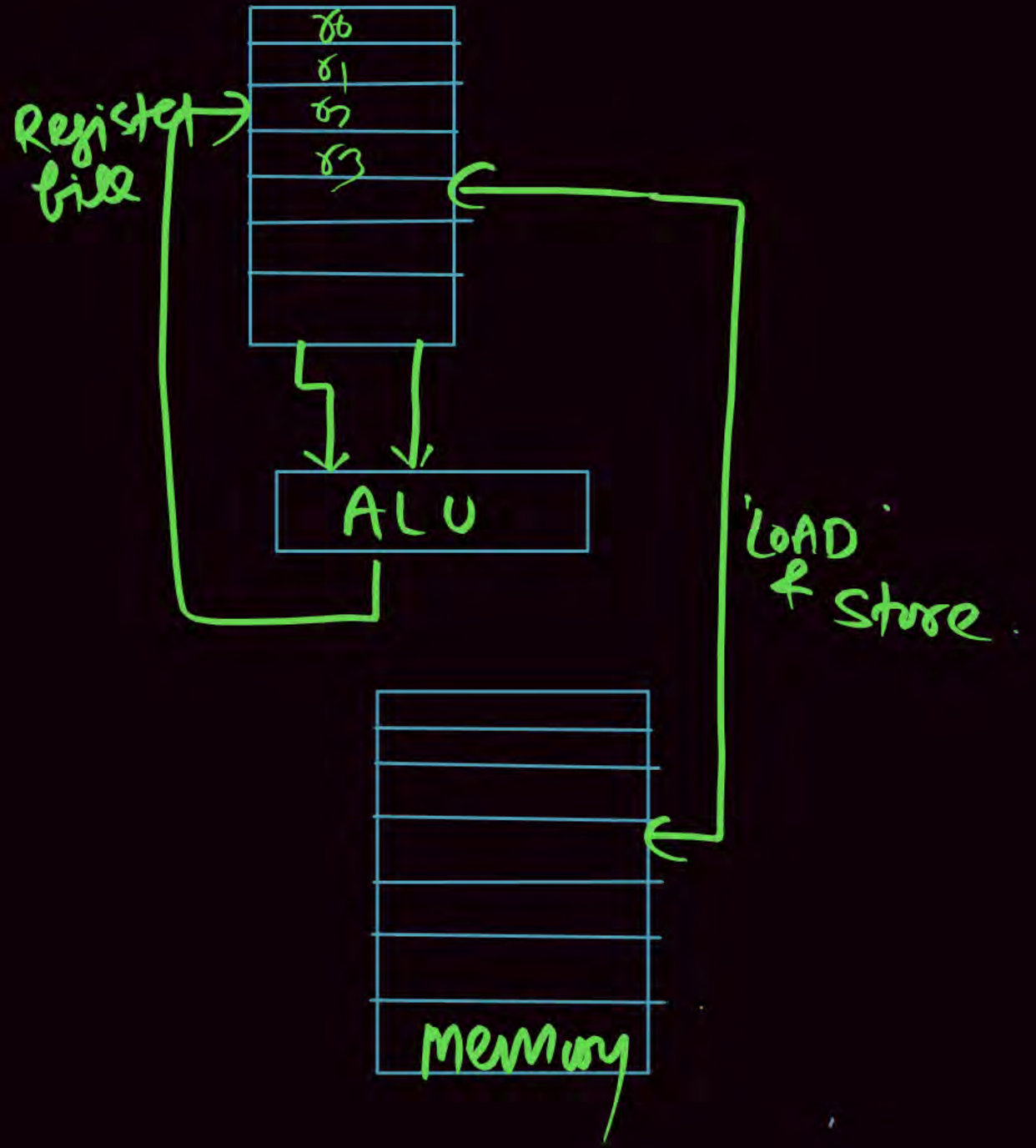
Register - Register Reference: This Arch. Supports More Number of Registers

Compitable Inst. Format

ALU operation | Destn Reg. | Source 1 Reg | Source 2 Reg

Note

In this org. ALU operand Always Required in Registers.


R0
R1
R2
R3
Register file
ALU
LOAD & Store
Memory

General Register Organization

Eg (A+B) Using Reg-Reg Reference CPU ?

I_1: LOAD γ_0 A; $\gamma_0 \in M[A]$

I_2: LOAD γ_1 B; $\gamma_1 \in M[B]$

I_3: ADD γ_2 γ_0 γ_1; $\gamma_2 \in \gamma_0 + \gamma_1$.

---

3 m/c Instn Using (Reg-Reg Ref.)

∴

General Register Organization

(3) $(x*y)+z$: Using Reg-Reg Ref ?

I_1: LOAD r_0 x; $r_0 \in M[x]$

I_2: LOAD r_1 y; $r_1 \in M[y]$

I_3: MUL r_2 r_0 r_1; $r_2 = r_0 * r_1$ $r_2 = x*y$

I_4: LOAD r_3 z; $r_3 \in M[y]$

I_5 ADD r_4 r_2 r_3; $r_4 = r_2 + r_3$ $r_4 = (x*y)+z$

5 m/c Instn using Reg. Reg CPU

General Register Organization

$X = (A+B) * (C+D)$

Here A, B, C, D & X are Variables in memory.
How many M/c Instn Required Using Reg-Reg CPU?

I_1: LOAD γ_0 A;

I_2: LOAD γ_1 B;

I_3: ADD γ_2 γ_0 γ_1; ($\gamma_2 = A+B$)

I_4: LOAD γ_3 C

I_5 LOAD γ_4 D

I_6 ADD γ_5 γ_3 γ_4 ($\gamma_5 = C+D$)

I_7 MUL γ_6 γ_2 γ_5 $\gamma_6 = (A+B) * (C+D)$

I_8 STORE X γ_6; $M[X] = (A+B) * (C+D)$

8 M/c Instn (Using Reg-Reg CPU)

GATE

(20) of 2 Marks

Q Minimum Number of Register ?

r_0, r_0, r_1

$r_0 \leftarrow r_0 + r_1$

Reg - Mem Ref.

General Register Organization

2 Marks GATE

$X = (A+B) * (C+D)$

Here A, B, C, D & X are Variables in memory.
How many Minimum # Registers Required using Reg-Reg CPU?

I_1: LOAD γ_0 A;

I_2: LOAD γ_1 B;

I_3: ADD γ_0 γ_0 γ_1 ; $\gamma_0 \in \gamma_0 + \gamma_1$ $\gamma_0 = A+B$

I_4: LOAD γ_1 C;

I_5: LOAD γ_2 D;

I_6: ADD γ_1 γ_1 γ_2; $\gamma_1 \in \gamma_1 + \gamma_2$. $\gamma_1 = C+D$

I_7: MUL γ_2 γ_0 γ_1 ; $\gamma_2 \in \gamma_0 * \gamma_1$; $\gamma_2 = (A+B) * (C+D)$

I_8: STORE X γ_2; M[X] $\in (A+B) * (C+D)$

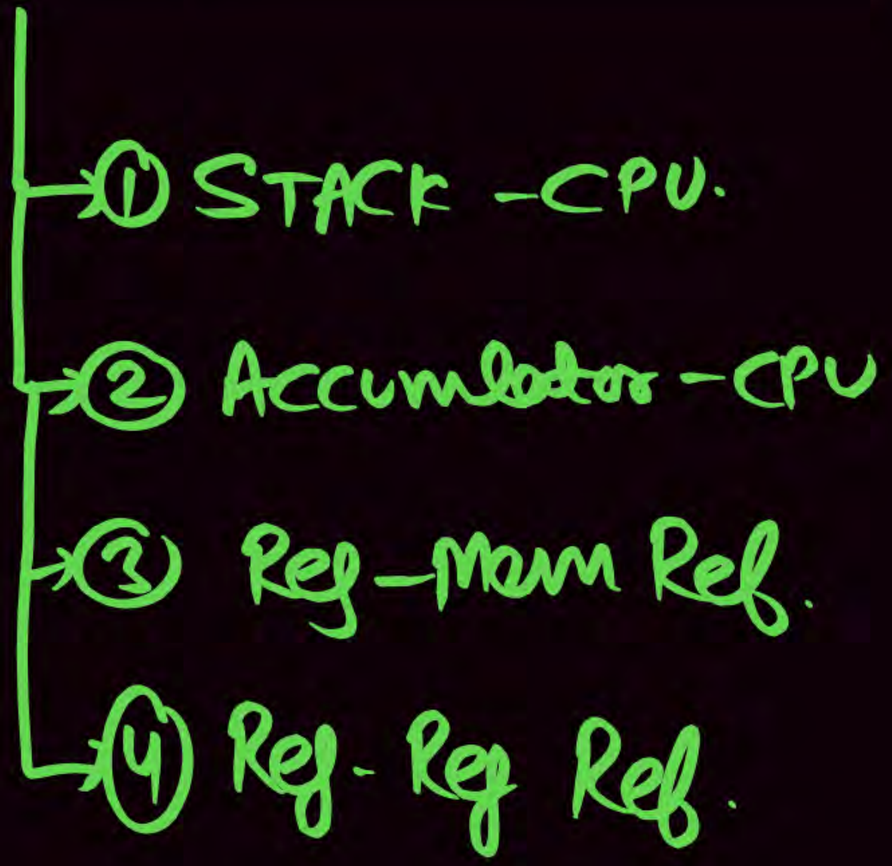
3 Minimum Reg required.

RISC Instructions

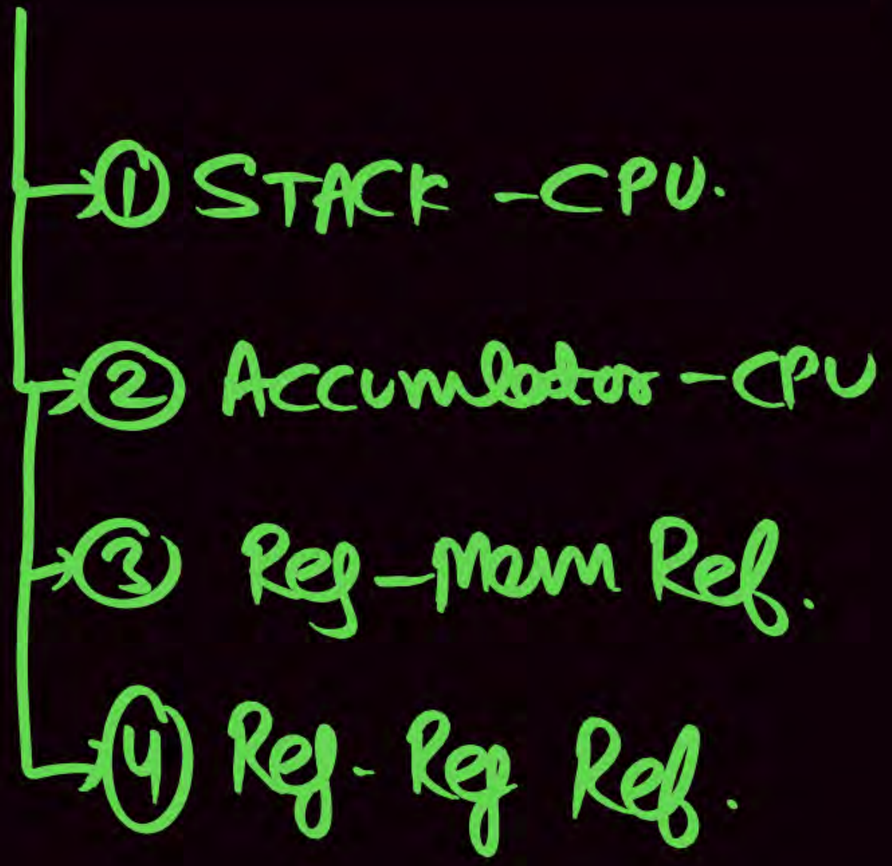
$$X = (A + B) \times (C + D)$$

LOAD	R1, A	R1 ← M[A]
LOAD	R2, B	R2 ← M[B]
LOAD	R3, C	R3 ← M[C]
LOAD	R4, D	R4 ← M[D]
ADD	R1, R1, R2	R1 ← R1 + R2
ADD	R3, R3, R2	R3 ← R3 + R4
MUL	R1, R1, R3	R1 ← R1 × R3
STORE	X, R1	M[X] ← R1

eg) (A+B)



Ex2 $(x*y)+z$.



Q3 $X = (A + B) * (C + D)$

- ① STACK - CPU.
- ② Accumulator - CPU
- ③ Reg - Mem Ref.
- ④ Reg - Reg Ref.

Note:

Immediate field is n bit

Unsigned Range = $(0 \text{ to } 2^n - 1)$

Signed Range = $-(2^{n-1})$ to $+ (2^{n-1} - 1)$

Example

If immediate field is 4 bit

Then unsigned range = $(0 \text{ to } 2^4 - 1) \Rightarrow 0 \text{ to } 15$

Signed Range = $-(2^{4-1})$ to $+ (2^{4-1} - 1)$

Q.

A machine has a 32-bit architecture, with 1-word long instructions. It has 64 registers, each of which is 32 bits long. It needs to support 45 instruction, which have an immediate operand in addition to two register operands. Assuming that the immediate operand is an unsigned integer the maximum value of the immediate operand is 16383 **[GATE-2014 (Set-1)]**

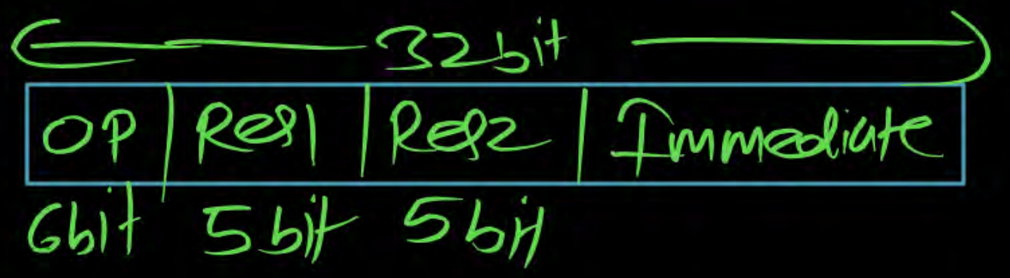
Done.

Q.

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 16 bit.

[GATE-2016 (Set-2)]

Done



Q.

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 a byte-aligned fashion. If a program has 100 instructions, the amount of memory (in bytes) consumed by the program text is 500B.

[GATE-2016 (Set-2): 2Marks]

Done

Basic Terms and Notation

The alphabet of computers, more precisely digital computers, consists of 0 and 1.

Each is called a *bit*, which stands for the binary digit.
The term *byte* is used to represent a group of 8 bits.
The term *word* is used to refer to a group of bytes that is processed simultaneously.

The exact number of bytes that constitute a word depends on the system, For example, in the Pentium, a word refers to four bytes or 32 bits. On the other hand, eight bytes are grouped into a word in the Itanium processor.

We use the abbreviation “b” for bits, “B” for bytes, and “W” for words.

Sometimes we also use *doubleword* and *quadword*. A doubleword has twice the number of bits as the word and the quadword has four times the number of bits in a word.

Bits in a word are usually ordered from right to left, as you would write digits in a decimal number. The rightmost bit is called the *least significant bit* (LSB), and the leftmost bit is called the *most significant bit* (MSB).

Byte Ordering

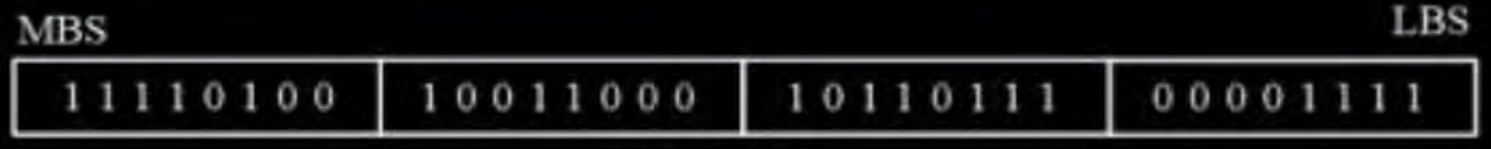
Storing data often requires more than a byte.

Suppose that we want to store these 4-byte data in memory at locations 100 through 103.

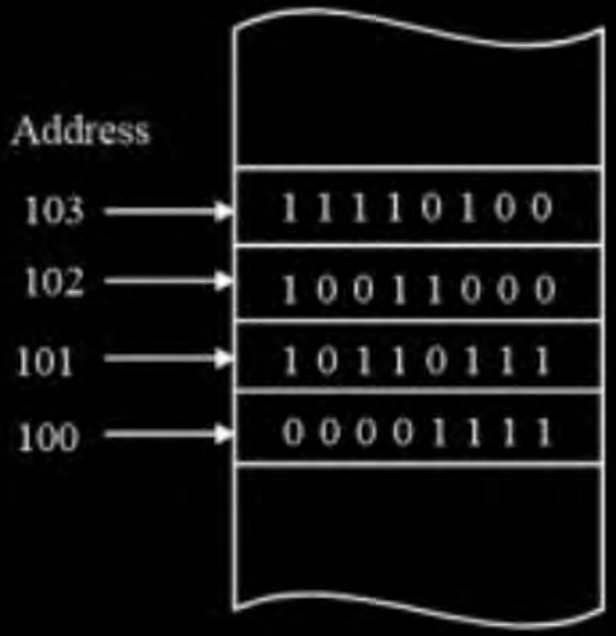
How do we store them?

Figure Shows two possibilities: Least significant byte or Most significant byte is stored at location 100. These two byte ordering schemes are referred to as the little endian and big endian.

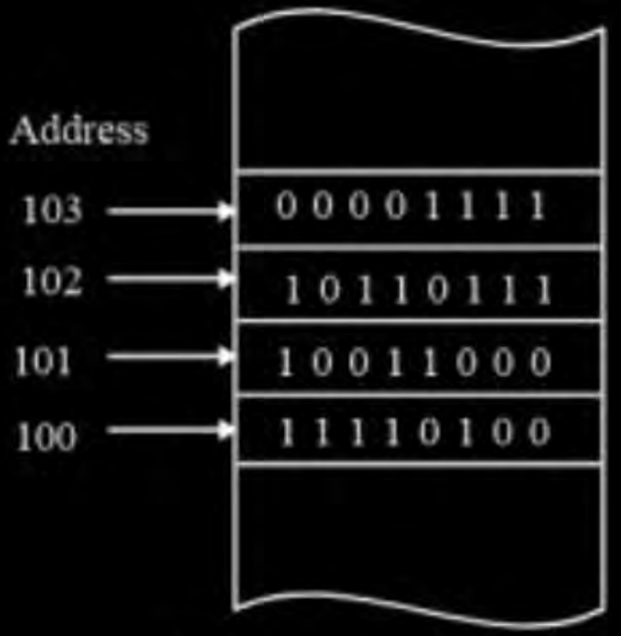
Two Important Memory Design Issues



(a) 32- bit data



(b) Little-endian byte ordering



(c) Big endian byte ordering

Two byte ordering schemes commonly used by computer systems.

Expand Opcode Technique

Expand Opcode Technique

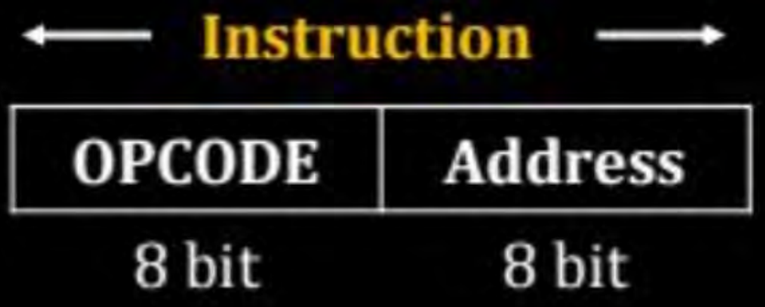
Expand opcode length is required in the fixed length instruction supported CPU Design to implement the various instruction with different formats.

Variable Length Instruction Supported CPU Design

OPCODE = 8 bit

Address field = 8 bit

(i) 1 Address Instruction Design:



= 16 bit

(ii) 0 Address Instruction Design:

= 8 bit [No need of expand opcode technique]

Fixed Length Instruction Supported CPU Design

OPCODE = 8 bit

A.F = 8 bit

(i) 1 Address Instruction Design:

= 16 bit

(ii) 0 Address Instruction Design:

= 16 bit [Here expand opcode technique required]

Expand Opcode Technique

- Primitive instruction means smallest opcode instruction.

Step 1: Identify the primitive instruction in the CPU.

Step 2: Calculate the total number of possible operation.

Step 3: Identify the free opcode after allocating the existed instruction

Step 4: Calculate the number of Derived instruction possible by multiply

$$\text{Free opcode} \times 2^{\text{Increment bit in opcode}}$$

Q.

Consider a processor which support 6 bit instruction and 4 bit address field. If there exist 2 one address instruction then how many 0 address instruction can be formulated?

Q.

Consider a processor which contain 8 bit word and 256 word memory. It support 3 word instruction. If these exist 254 2-address instruction and 256 1-address instruction then how many 0 address instruction can be formulated?

Q.

Consider a processor with 16 bit instruction. Processor has 15 registers and support 2 address instruction and 1 address instruction. If processor support 256 1-address instruction then number of 2-address instruction are

A 128

B 192

C 240

D 248

Solution(c): 240

15 register $= 2^4 \Rightarrow$ Register A.F = 4 bit

OPCDE field $= 16 - (4 + 4) = 8$ bit

So total number of 2 address instruction $= 2^8 = 256$

Let 'x' 2 address instruction used

Number of free opcode $= (2^8 - x)$

1 Address field

OPCODE	A.F
12 bit	4 bit

Total no of 1 address instruction = $(2^8 - x) \times 2^{12-8}$

$[2^8]256 \Rightarrow (2^8 - x) \times 2^4$

$2^4 = 2^8 - x$

$x = 2^8 - 2^4 \Rightarrow 256 - 16$

$= 240$

Q.

A processor has 16 register (R0, R1, ..., R15) and 64 floating point registers (F0, F1, ... F63). It uses a 2-byte instruction format. There are four categories of instructions: Type-1 Tpye-2 Type-3 and Type-4. Type-1 category consists of four instructions, each with 3 integer register operands (3Rs) Type-2 category consists of eight instructions, each with 2 floating point register operands (2fs). Type-3 category consist of fourteen instructions, each with one integer register operand and one floating point register operand (1R + IF). Type-4 category consists of N instructions, each with a floating point register operand (FR). The maximum value of N is _____.

[GATE-2018 : 2 Marks]

Q.

A processor has 64 registers and uses 16-bit instruction format. It has two types of instructions: I-type and R-type. Each I-type instruction contains an opcode, a register name, and a 4-bit immediate value. Each R-type instruction contains an opcode and two register names. If there are 8 distinct I-type opcodes, then the maximum number of distinct R-type opcodes is _____.

[GATE-2020 : 2 Marks]

THANK YOU!