COMPUTER SCIENCE

Computer Organization and Architecture

Machine Instruction and Addressing Modes

Machine Instruction

Lecture_03

Vijay Agarwal sir

01 Instruction Set Architecture

02 Expand Opcode Technique

Instruction

OPERAND ⇒ DATA.

AF

Opcode ⇒ operational Code.

↳ Type of operation.

Memory LMB = 2^{20} Byte

So Mem AF = 20 bit.

n bit opcode Can perform 2^n operation.

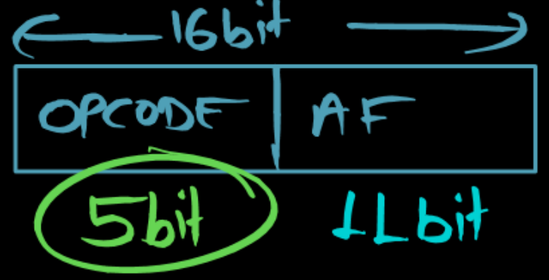
(Q) 2 bit opcode

00 → ADD
01 → OR
10 → AND
11 → MUL

2^2 = 4 operation.

Q) a 16 bit Instn placed in 2KB memory having only mem. AF then How many Total # operation can be supported

Soln



$2KB = 2^1 2^{10} B = 2^{11}$ Byte

A.F = 11 bit

Total # operation $= 2^5 = 32$ operation.

Min # operation = 1

max # operation = 32.

Instruction Format

→ Next Instⁿ Address But for that we have PC Register so 4AI Not Used.

Reg-CPU

ADD R_1 R_2 R_3; $R_1 \leftarrow R_2 + R_3$;

ADD R_1 R_2; $R_1 \leftarrow R_1 + R_2$

AC-CPU

ADD R_1 $AC \leftarrow AC + R_1$

STACK-CPU

ADD Stack.

Machine Instruction Characteristics

- The operation of the processor is determined by the instructions it executes, referred to as machine instructions or computer instructions
- The collection of different instruction that the processor can execute is referred to as the processor's instruction set **(ISA)**

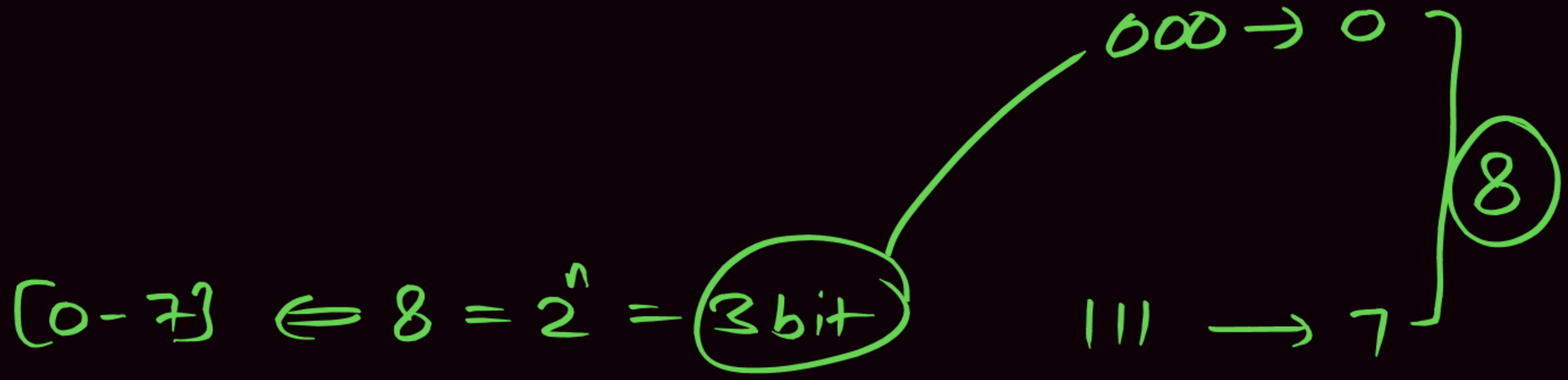
Instruction Set of 20.

20 distinct operation/Instruction.

OPCODE = 5 bit

$2^4 = 16$

5 bit $= 2^5 = 32$ (17-32)



$16 = 2^n$ = 4bit

0000 → 0

16

1111 → 15

Memory starts from '0'

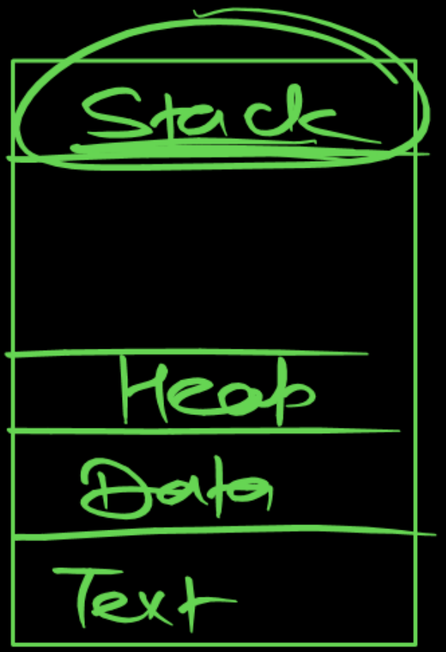
n bit A.L Represent 2^n Memory cells.

Range (0 to $2^n - 1$)

Instruction Format

Operating System

Process Concept



Process in Memory.

Instruction Set Architecture Classification

1. Stack organization.
2. Single accumulator organization.
3. General register organization.

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

- In Stack Based org ALU operation are performed only on Stack Data.
- So Both the operand (DATA) Must be Present (Required) in the Stack & After the Processing Result is also Stored in the Stack.
- Stack is a part of Memory which is initialized by (Stack Pointer) SP Register.
  In the Stack Insert & Delete operation are take place at same [One] end Called Top of the Stack [TOS], so it become Default Location. LIFO

Stack Organization

Combitable Instⁿ format

OPCODE

Type of operation

Destination Source 1 Source 2

ALU operation ⇒ ADD

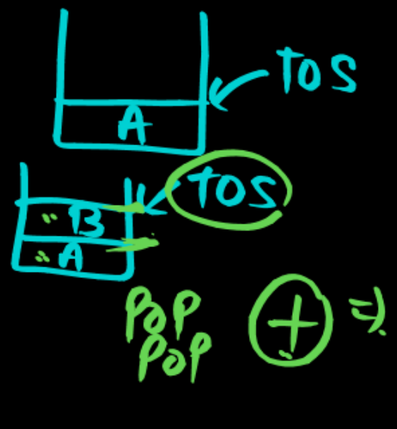
TOS ← TOS + TOS

eg (A+B)

① PUSH A

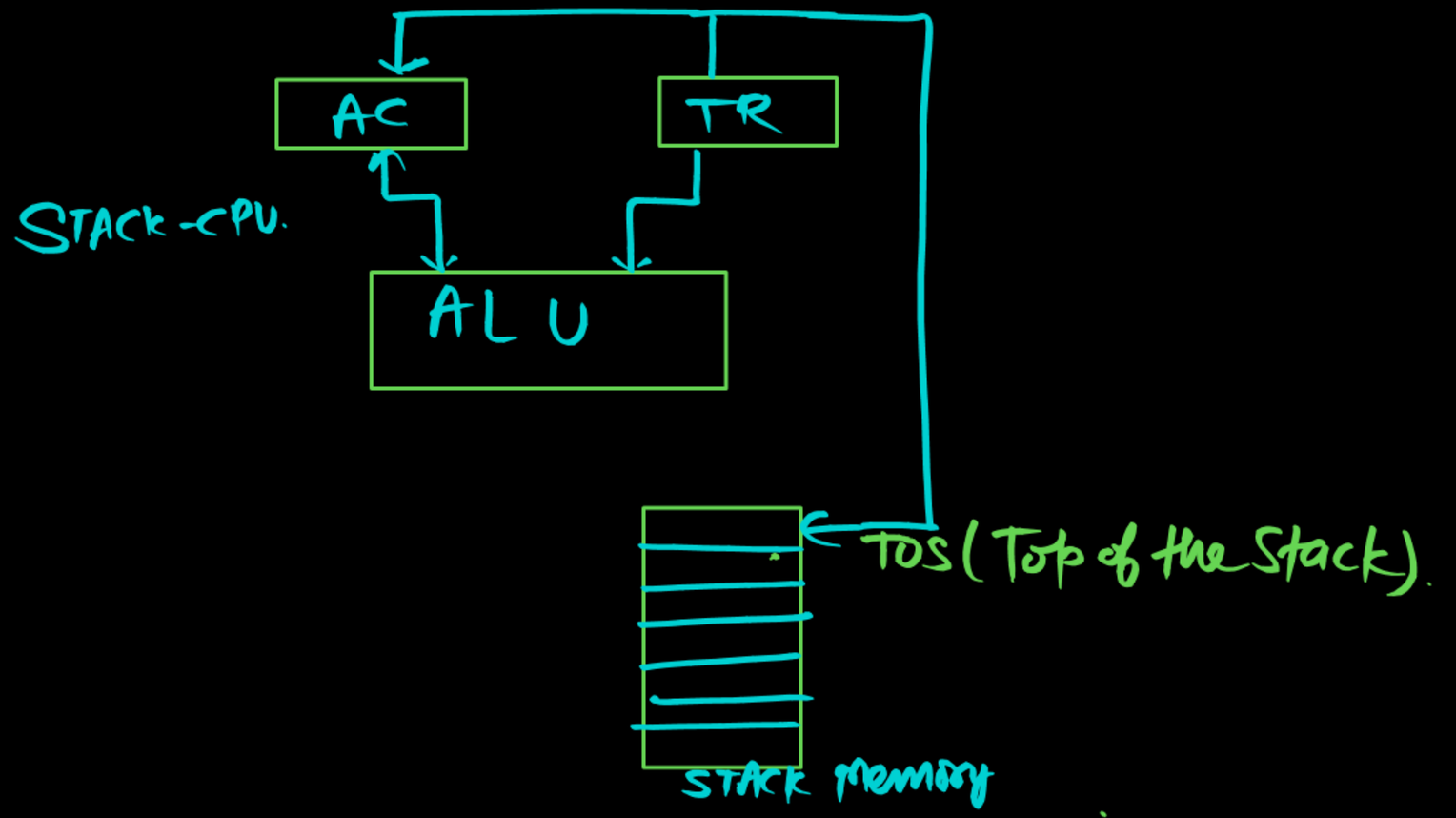
② PUSH B

③ ADD



3 mk Instⁿ (STACK-CPU)

Stack Organization



Stack Organization

eg $(x * y) + z$

How Many Machine Instn Required to execute Using STACK - CPU ?

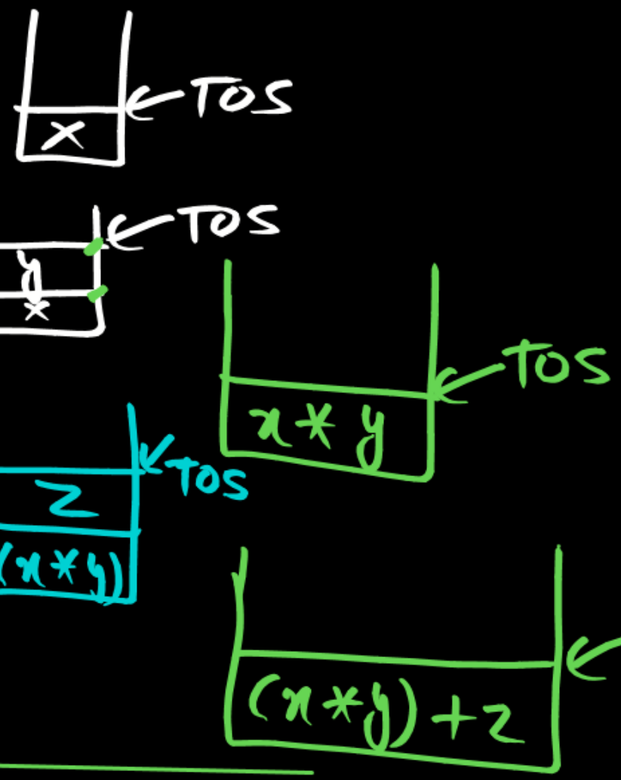
I_1 PUSH X

I_2 PUSH Y

I_3 MUL

I_4 PUSH Z

I_5 ADD



5 M/c Instn Using (STACK - CPU)

Stack Organization

In Stack Based org (Stack-CPU) only ALU operation are 0 Address Instruction & Data transfer Instn/operation [PUSH & POP] are Not 0AI [0 Address Instruction].

Stack Organization

Q. Consider a 32 bit Hypothetical Processor which use STACK-CPU. Which support 1 Word opcode and 24 bit address following statement is executed on a STACK-CPU (Stack is Initially Empty)

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

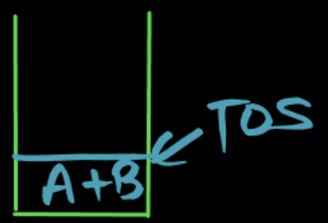
Q.(i)

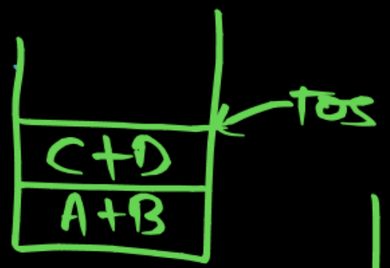
How many Machine Instruction are required using Stack-CPU?

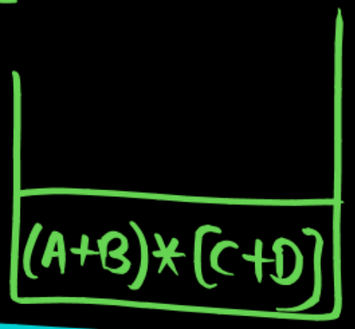
$$X = (A+B) * (C+D)$$

I_1 : PUSH A
I_2 : PUSH B
I_3 ADD
I_4 PUSH C
I_5 PUSH D
I_6 ADD
I_7 MUL
I_8 POP X;

8 m/c Instn [STACK-CPU].







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

Q.(ii) How much Memory space is required for the program in Byte?

Soln

32 bit processor.

1 Word = Opcode

Opcode Size = 1 Word
= 32 bit

OPCODE = 4 Byte

Address = 24 bit

A.F = 3 Byte

I_1	PUSH A	4B + 3B	= 7B
I_2	PUSH B	4B + 3B	= 7B
I_3	ADD	4B	= 4B
I_4	PUSH C	4B + 3B	= 7B
I_5	PUSH D	4B + 3B	= 7B
I_6	ADD	4B	= 4B
I_7	MUL	4B	= 4B
I_8	POP X	4B + 3B	= 7B

47 Byte Ans

Q.(iv)

What is the status of the Stack at the end of program execution?

Soln

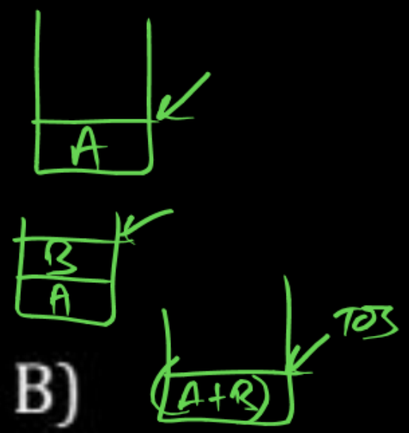
Empty

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

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

- In the Accumulator Based, first ALU operand Required in the Accumulator, second ALU operand either Present in Register (or) Present in Memory.
- After the Processing Result is stored in Accumulator (or) Accumulator is used as Destination.
- In the Processor Only 1 Accumulator is Present so No Need to explicit Mention the Address.

Single Accumulator Organization

Instn format:

Type of operation

(i) ALU operation :

Destination	Source 1	Source 2
AC	AC	Reg / Mem

(ii) Data transfer operation

Destination	Source	
AC	Mem	[Memory Read] LOAD
Mem	AC	[Memory Write] STORE

ALU operation.

(i) ADD R_L

$AC \leftarrow AC + R_L$

(Destn) $[S_1]$ (AC) $[S_2]$ Reg

(2) ADD [6000]

$AC \leftarrow AC + M[6000]$

(Destn) AC; S_1 (AC); S_2 (MEM)

Data transfer operation.

(1) LOAD [7000]

$AC \leftarrow M[7000]$

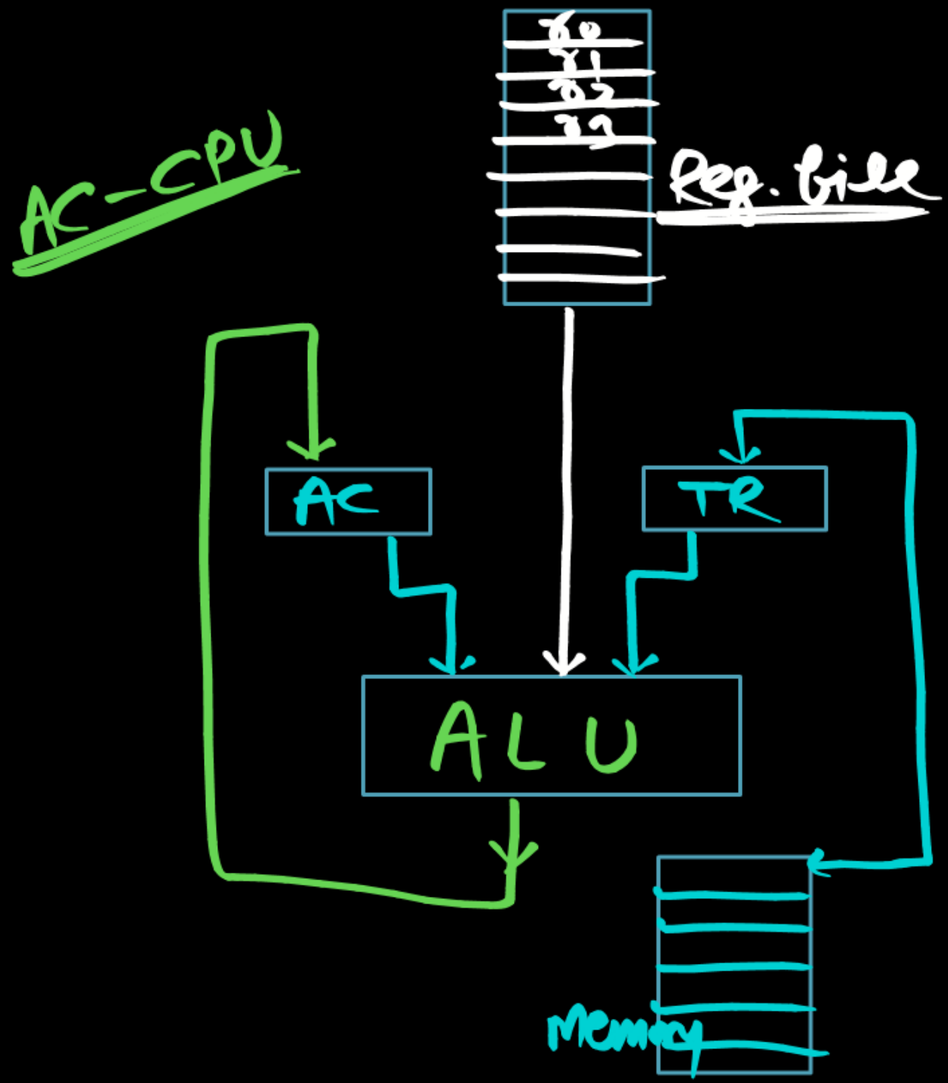
Destn (AC); Source (Mem)

(2) STORE [9000]

$M[9000] \leftarrow AC$

Destn (Mem); Source (AC).

Single Accumulator Organization



Single Accumulator Organization

eg1 (A + B) A & B Variable in the memory. # M/c Instn Using AC - CPU ?

I_1 : LOAD A; AC ∈ M[A]

I_2 : ADD B; AC ∈ AC + M[B]

2 Machine Instn [AC - CPU].

eg2. (x * y) + z Using AC-CPU.?

I_1 LOAD X; AC ∈ M[X]

I_2 MUL y; AC ∈ AC * M[y]

I_3 ADD z; AC ∈ AC + M[z]

3 M/c Instn [Using AC - CPU].

(eg) $X = (A+B) * [C+D]$ A, B, C, D & X are the variable in the memory

(i) How many Machine Instruction (m/c Instⁿ) are required to evaluate using AC-CPU org.?

(ii) How many Spills are Required?

I_1 :	LOAD	A;	$AC \leftarrow M[A]$
I_2 :	ADD	B;	$AC \leftarrow AC + M[B]$
I_3 :	STORE	T;	$M[T] \leftarrow AC$
I_4 :	LOAD	C;	$AC \leftarrow M[C]$
I_5 :	ADD	D;	$AC \leftarrow AC + M[D]$
I_6 :	MUL	T;	$AC \leftarrow AC * M[T]$
I_7 :	STORE	X;	$M[X] \leftarrow AC$

7 m/c Instⁿ (AC-CPU)

To store intermediate Result in memory.

$M[T] = (A+B)$ → Spills

(i) 7 m/c Instⁿ (AC-CPU)

(ii) 1 Spill Required.

Single Accumulator Organization

X = (A + B) × (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

(Register file)

Based on the Number of Register supported by the Processor this Arch. is Divided into 2 Type.

- Memory
  1. Register – Mem Ref.
  2. Reg – Register Ref.
- Register file.
  (The Number of Register supported by the Processor Denoted by Register file.

General Register Organization

① Register – memory Ref.

This Arch. Support less Number of Register, therefore Register file Size Small.

Instn Format

ALU operation

Data transfer Oper.

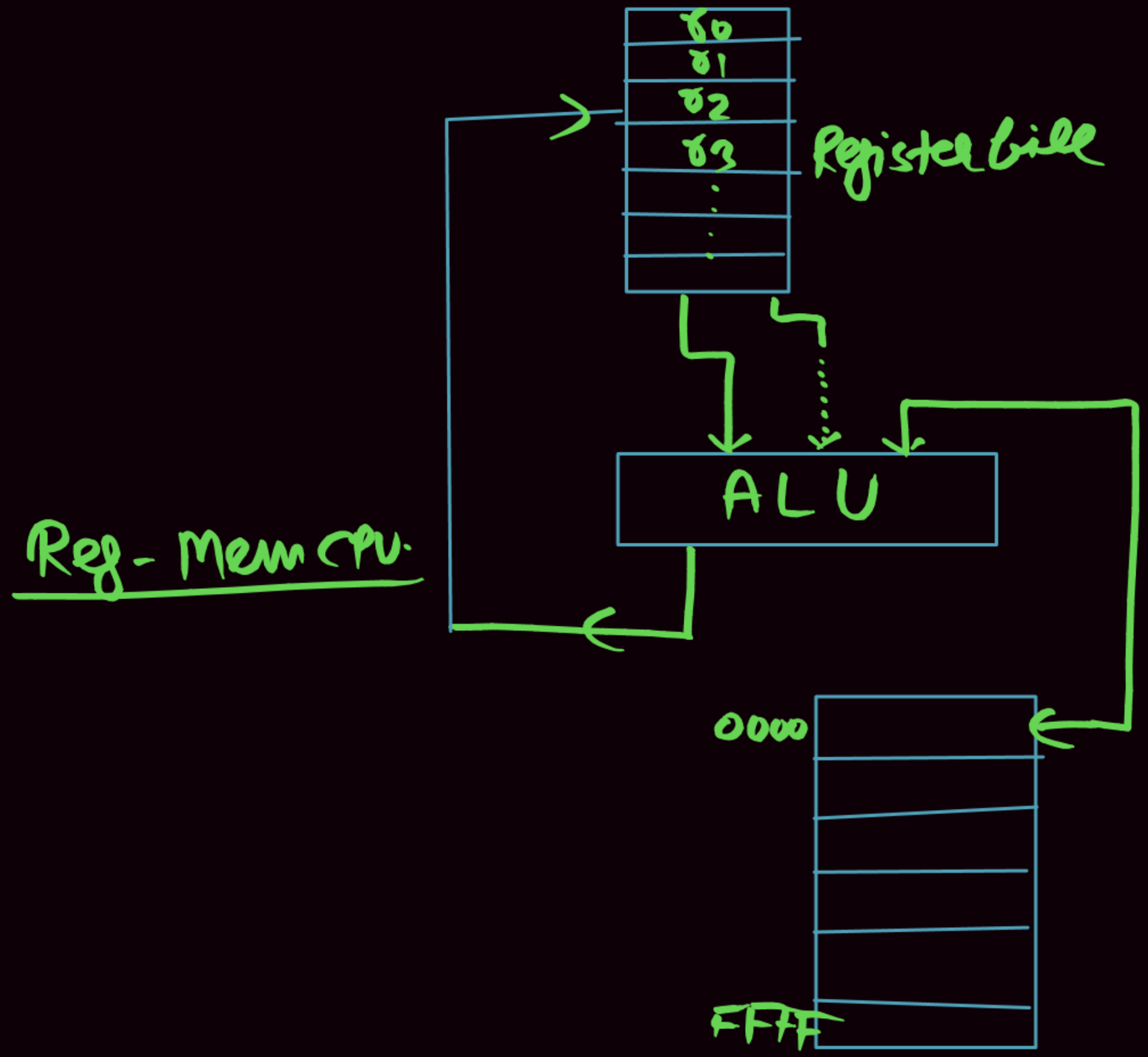
Dest^n — Reg [S1]

Source 1 — Reg

Source 2 — Reg/Mem

Mem, Reg, Reg

Reg, Reg, Mem } 'Mov'


R0
R1
R2
R3
Register bill
ALU
Reg - Mem CPU.
0000
FFFF

(eg) $(x*y)+z$. Reg − mem Rel.

I_1: MOV r_0 X; $r_0 \in M[x]$

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

I_3: ADD r_0 z; $r_0 \leftarrow r_0 + M[z]$

3 m/c Instn (Reg − Mem Rel)

(eg) A+B. Using Reg-Mem.

I_1 MOV r_0 A; $r_0 \in M[A]$

I_2 ADD r_0 B; $r_0 \leftarrow r_0 + M[B]$

2 M/c Instn (Reg-Mem).

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

A, B, C, D & X are Variable in the memory
How many Instn using Reg – Mem CPU?

I_1 : MOV γ_0 A; $\quad \gamma_0 \in M[A]$

I_2 : ADD γ_0 B; $\quad \gamma_0 \in \gamma_0 + M[B]$.

I_3 MOV γ_1 C; $\quad \gamma_1 \in M[C]$

I_4 ADD γ_1 D; $\quad \gamma_1 \in \gamma_1 + M[D]$

I_5 MUL γ_0 γ_1; $\quad \gamma_0 \in \gamma_0 * \gamma_1$

I_6 MOV X, γ_0 $\quad M[X] \in \gamma_0$

---

6 mc Instr (Reg – Mem Ref).

General Register Organization

Reg - Mem Ref:

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

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)

Reg - Reg Ref CPU. (RISC)

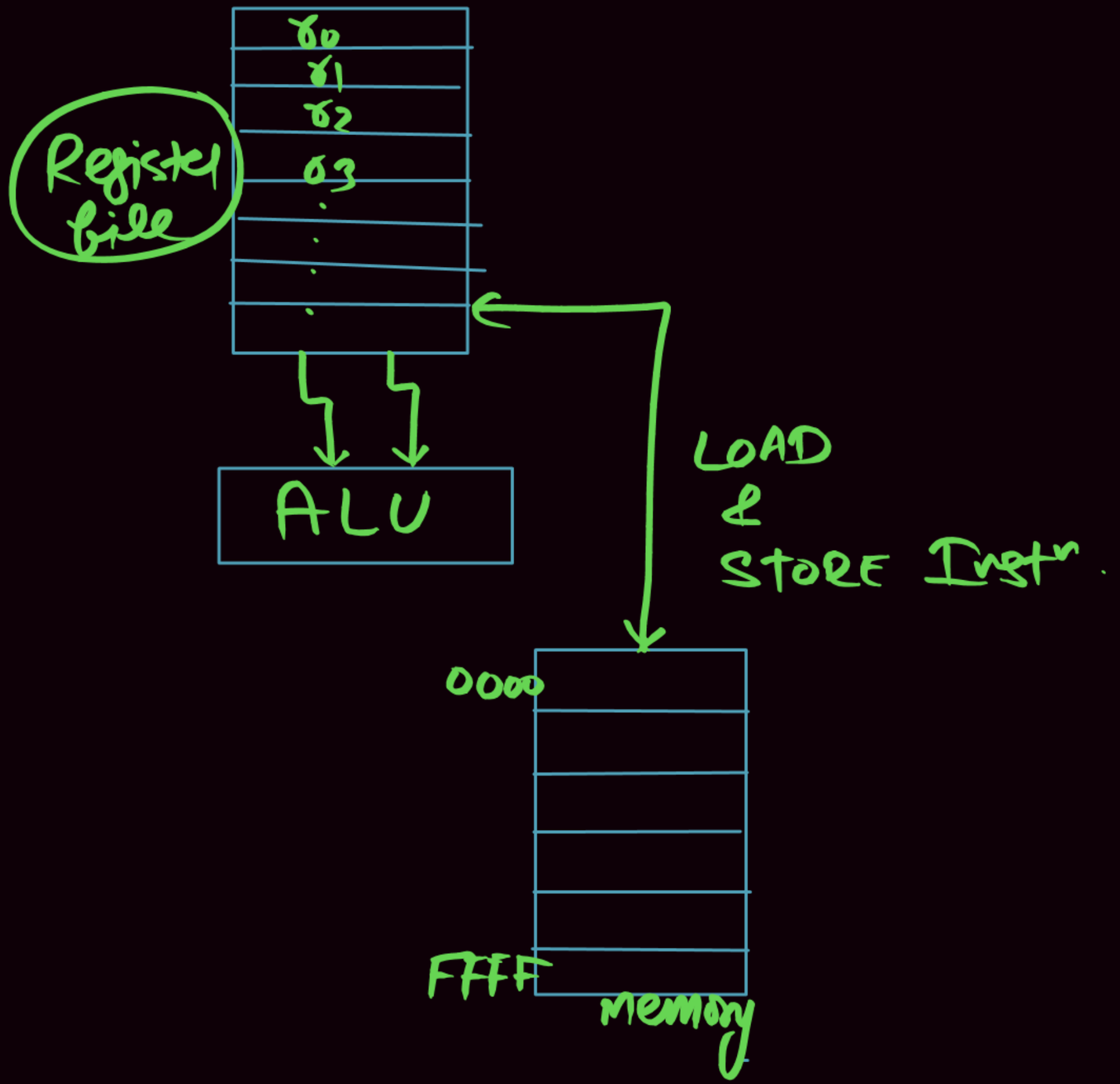
In this Arch. it Support More Number of Register.

Instn Format

ALU operation:

Destn	Source 1	Source 2
Reg	Reg	Reg

Note In this Arch ALU operand always Required in the Register.


R0
R1
R2
R3
Register bill
ALU
LOAD
&
STORE Instn.
0000
FFFF
memory

(eg) $(x * y) + z$; How Many M/C Instn Using Reg-Reg Rel.

I_1: LOAD γ_0 X; $\gamma_0 \in M[x]$

I_2: LOAD γ_1 y: $\gamma_1 \in M[y]$

I_3 MUL γ_2 γ_0 γ_1; $\gamma_2 \in \gamma_0 * \gamma_1$

I_4 LOAD γ_3 z; $\gamma_3 \in M[z]$.

I_5 ADD γ_4 γ_2 γ_3; $\gamma_4 \in \gamma_2 + \gamma_3$

---

5 M/c Instn (Reg. Reg CPU).

(eg) (A + B) Using (Reg-Reg)

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).

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

Reg-Reg CPU ?

RISC Instructions

X = (A + B) × (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

THANK YOU!