Central Processing Unit

General Register Organization

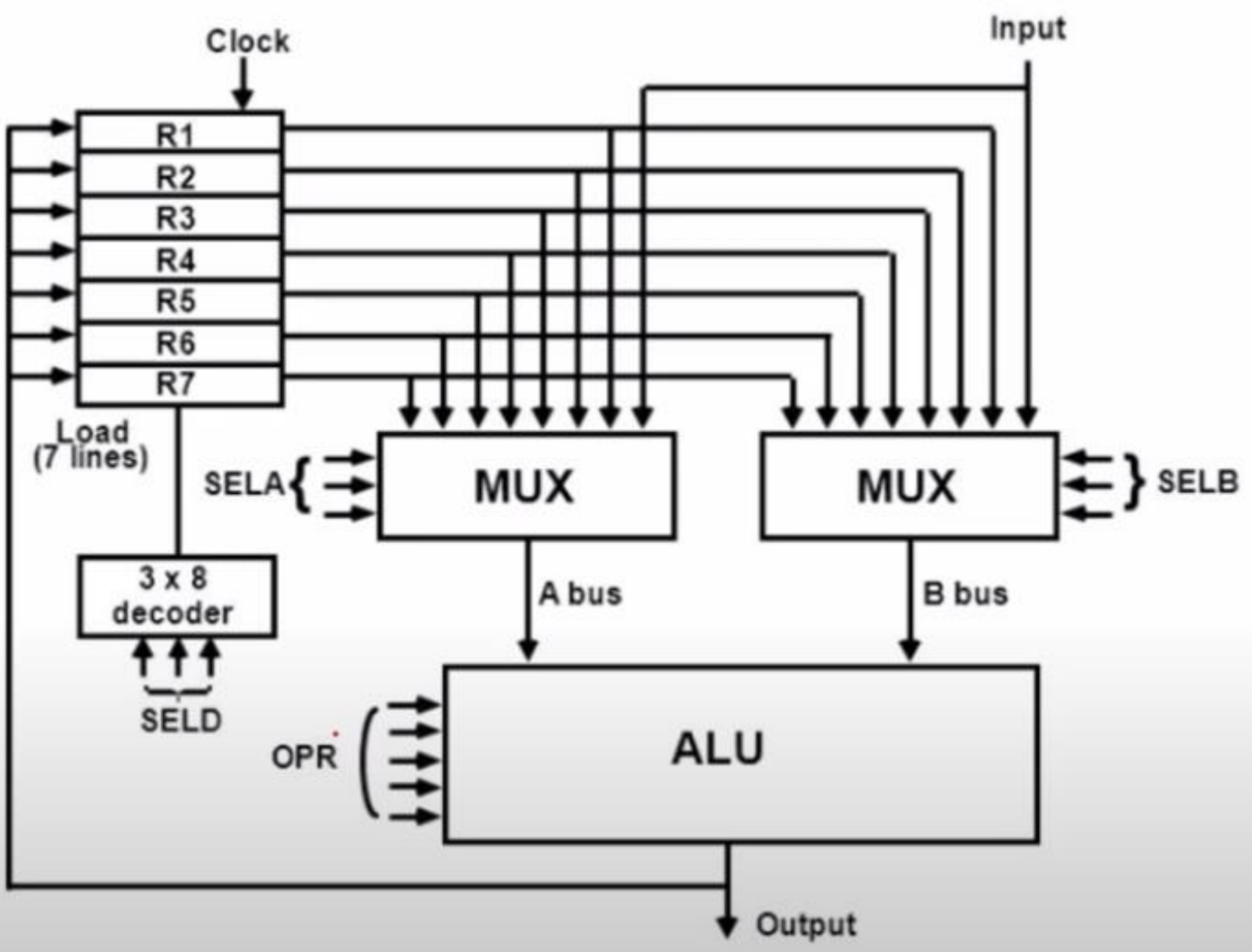
- A set of flip-flops forms a register
- A register is a unique high-speed storage area in the CPU

Registers implement two important functions in the CPU operation:

- It can support a temporary storage location for data and fast access to the data if required
- It can save the status of the CPU and data about the implementing program

- Generally CPU has **seven** general registers
- Register organization **show** how registers are selected and how **data flow** between register and ALU

Register set with common ALU



A general organization of seven CPU registers

Example

To perform the operation ***R3 = R1+R2*** We need to provide following binary selection variable to the select inputs

- **MUX A Selector (SELA):** **001** -To place the contents of R1 into bus A
- **MUX B Selector (SELB):** **010** - to place the contents of R2 into bus B
- **ALU Operation Selector (OPR):** **10010** – to perform the arithmetic addition A+B
- **Decoder Destination Selector (SEL D):** **011** – to place the result available on output bus in R3

Control Word

- The combined value of a binary selection inputs specifies the control word
- Control words for all micro operation are stored in the control memory
- It consist of four fields SELA, SELB and SELD contains three bit each and OPR field contains five bits
- Therefore, the **total bits** in the control word are 14

3	3	3	5
SEL A	SEL B	SEL D	OPR

FORMATE OF CONTROL WORD

- 3-bits of SELA select a source registers of the A input
- 3-bits of SELB select a source registers of the B input
- 3-bits of SELD or SELREG select a destination register using the decoder
- 5-bits of OPR select the operation to be performed by ALU

Table: Encoding of register selection fields

CHITKARA UNIVERSITY

Binary Code	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Encoding of ALU operations

Encoding of ALU Operations

OPR Select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A + B ✓	ADD
00101	Subtract A - B	SUB
00110	Decrement A	DECA
01000	ADD A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLA

Examples of microoperations for CPU

CHITKARA UNIVERSITY

ALU Micro-operations

Micro-operation	SELA	SELB	SELD	OPR	Control Word			
R1 ← R2 – R3	R2	R3	R1	SUB	010	011	001	00101
R4 ← R4 V R5	R4	R5	R4	OR	100	101	100	01010
R6 ← R6 + R1	-	R6	R1	INCA	110	000	110	00001
R7 ← R1	R1	-	R7	TSFA	001	000	111	00000
Output ← R2	R2	–	None	TSFA	010	000	000	00000
Output ← Input	Input	-	None	TSFA	000	000	000	00000
R4 ← shl R4	R4	-	R4	SHLA	100	000	100	11000
R5 ← 0	R5	R5	R5	XOR	101	101	101	01100

Micro-operation	SELA	SELB	SELD	OPR
R1 ← R2 – R3	R2	R3	R1	SUB
R4 ← R4 ∨ R5	R4	R5	R4	OR
R6 ← R6 + R1	-	R6	R1	INCA
R7 ← R1	R1	-	R7	TSFA
Output ← R2	R2	–	None	TSFA
Output ← Input	Input	-	None	TSFA
R4 ← shl R4	R4	-	R4	SHLA
R5 ← 0	R5	R5	R5	XOR

Stack Organization

CHITKARA UNIVERSITY

Stack Organization

- ❑ **Stack** is a storage structure that stores information in such a way that the last item stored is the first item retrieved
- ❑ It is based on the **principle** of **LIFO** (Last-in-first-out)
- ❑ The stack in digital computers is a group of memory locations with a register that holds the address of top of element
- ❑ This register that holds the address of top of element of the stack is called **Stack Pointer**

- In Stack organization, ALU operations are performed on stack data i.e. both the operands are always required on the stack
- After manipulation, the result is placed in the stack

Stack Operations

The main **two operations** that are performed on the operators of the stack are :

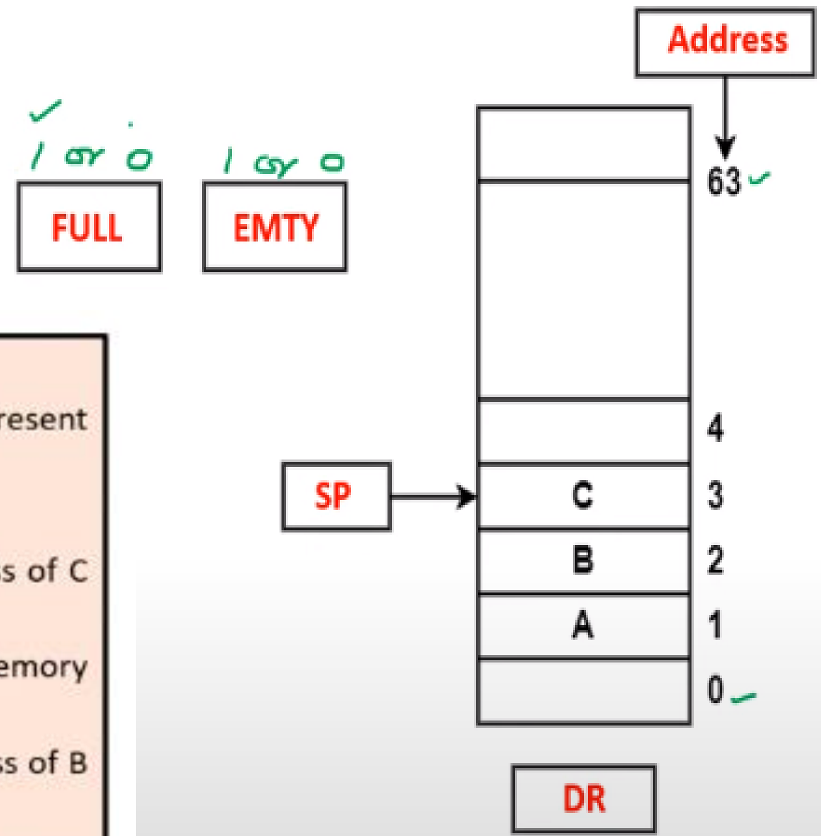
- **Push:** Insert an item on top of stack (decrementing the SP register)
- **Pop:** Delete an item from top of stack (Incrementing the SP register)

Implementation of Stack

In digital computers, stack can be **implemented** in two ways:

- Register Stack
- Memory Stack

A **stack** can be organized as a collection of **finite number of registers** that are used to store temporary information during the execution of a program



Block Diagram of 64-word Stack

- Fig shows the 64-word register stack arrangement
- The stack pointer register holds the address of the element present at the top of the stack
- Three-element A, B, and C are located in the stack
- Element C is at the top of the stack and SP holds the address of C i.e. 3
- The top element is popped from the stack through reading memory word at address 3 and decrementing the SP by 1
- Then, B is at the top of the stack and the SP holds the address of B that is 2
- It can insert a new word, the stack is pushed by incrementing the stack pointer by 1 and inserting a word in that incremented location

DR=Data Register

The PUSH operation is executed as follows: Insert an item on top of stack

SP←SP + 1	It can increment stack pointer
M[SP] ← DR	It can write element on top of the stack
If (SP = 0) then (FULL ← 1)	Check if stack is full
EMTY ← 0	Mark the stack not empty

FULL

$64 \rightarrow 2^6$

32 16 8 4 2 1

$\underbrace{1\ 1\ 1\ 1\ 1\ 1}$ → binary

+1

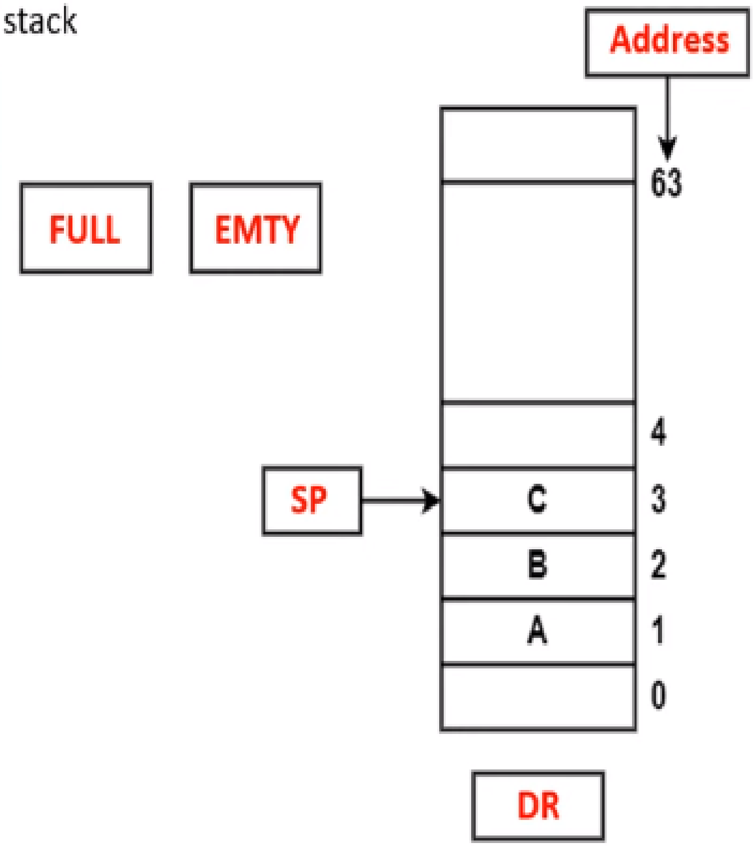
$\underbrace{1\ 0\ 0\ 0\ 0\ 0\ 0}$ ← SP 0

The PUSH operation is executed as follows:

- The stack pointer includes 6 bits, because $2^6 = 64$
- SP cannot exceed 63 (111111 in binary)
- After all, if 63 is incremented by 1, therefore the result is 0(111111 + 1 = 1000000). SP holds only the six least significant bits. If 000000 is decremented by 1 thus the result is 111111.
- Therefore, when the stack is full, the one-bit register 'FULL' is set to 1. If the stack is null, then the one-bit register 'EMTY' is set to 1
- The data register DR holds the binary information which is composed into or readout of the stack
- **First, the SP is set to 0, EMTY is set to 1, and FULL is set to 0.**
- Now, as the stack is not full (FULL = 0), a new element is inserted using the push operation.

The POP operation is executed as follows: Delete an item from top of stack

DR←M[SP]	It can read an element from the top of the stack
SP ← SP – 1	It can decrement the stack pointer
If (SP = 0) then (EMTY ← 1)	Check if stack is empty
FULL ← 0	Mark the stack not full



Block Diagram of 64-word Stack