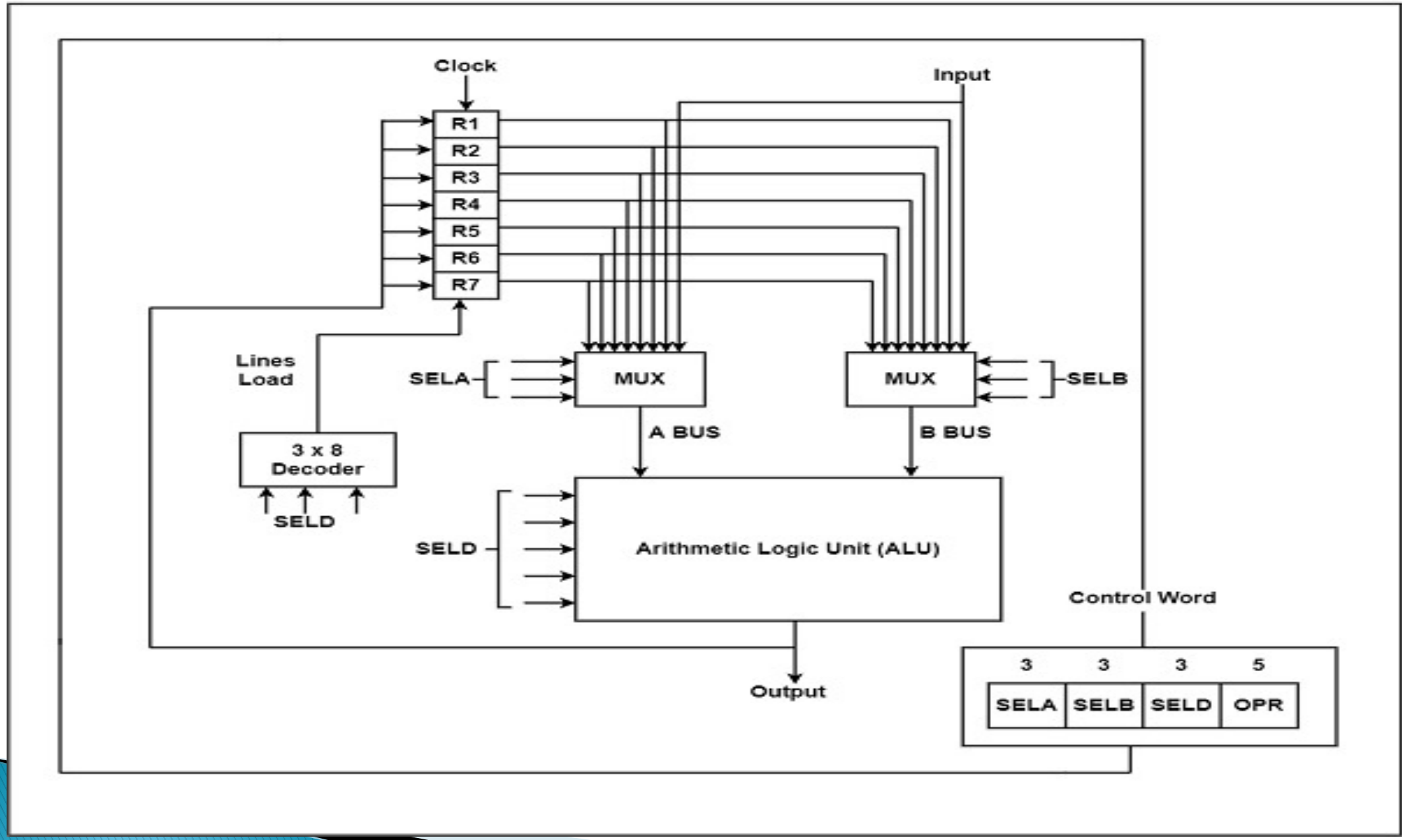
GENERAL REGISTER AND STACK ORGANIZATION

LECTURER: Dalia Adil

General Register Organization

- A register is a unique high-speed storage area in the CPU. They include combinational circuits that implement data processing. The information is always defined in a register before processing. The registers speed up the implementation of programs.
- Registers implement two important functions in the CPU operation are as follows −
- It can support a temporary storage location for data. This supports the directly implementing programs to have fast access to the data if required.
- It can save the status of the CPU and data about the directly implementing program.
- Example − Address of the next program instruction, signals get from the external devices and error messages, and including different data is saved in the registers.
- If a CPU includes some registers, therefore a common bus can link these registers. A general organization of seven CPU registers is displayed in the figure.

General Organization of Registers



- The CPU bus system is managed by the control unit. The control unit explicit the data flow through the ALU by choosing the function of the ALU and components of the system.
- **Consider R1 ← R2 + R3**, the following are the functions implemented within the CPU −
- **MUX A Selector (SELA)** − It can place R2 into bus A.
- **MUX B Selector (SELB)** − It can place R3 into bus B.
- **ALU Operation Selector (OPR)** − It can select the arithmetic addition (ADD).
- **Decoder Destination Selector (SELD)** − It can transfers the result into R1.
- The multiplexers of 3-state gates are performed with the buses. The state of 14 binary selection inputs determines the control word. The 14-bit control word defines a micro-operation..
- Control word

3	3	3	5
SELA	SELB	SELD	OPR

- The encoding of register selection fields is specified in the table

Encoding of Register Selection Field

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

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

ALU Micro-Operations

Micro-operation	SELA	SELB	SELD	OPR	Control Word			
R1 ← R2 - R3	R2	R3	R1	SUB	010	011	001	00101
R4 ← R4 ∨ R5	R4	R5	R4	OR	100	101	100	01010
R6 ← R6 + 1	R6	-	R6	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

Stack organization

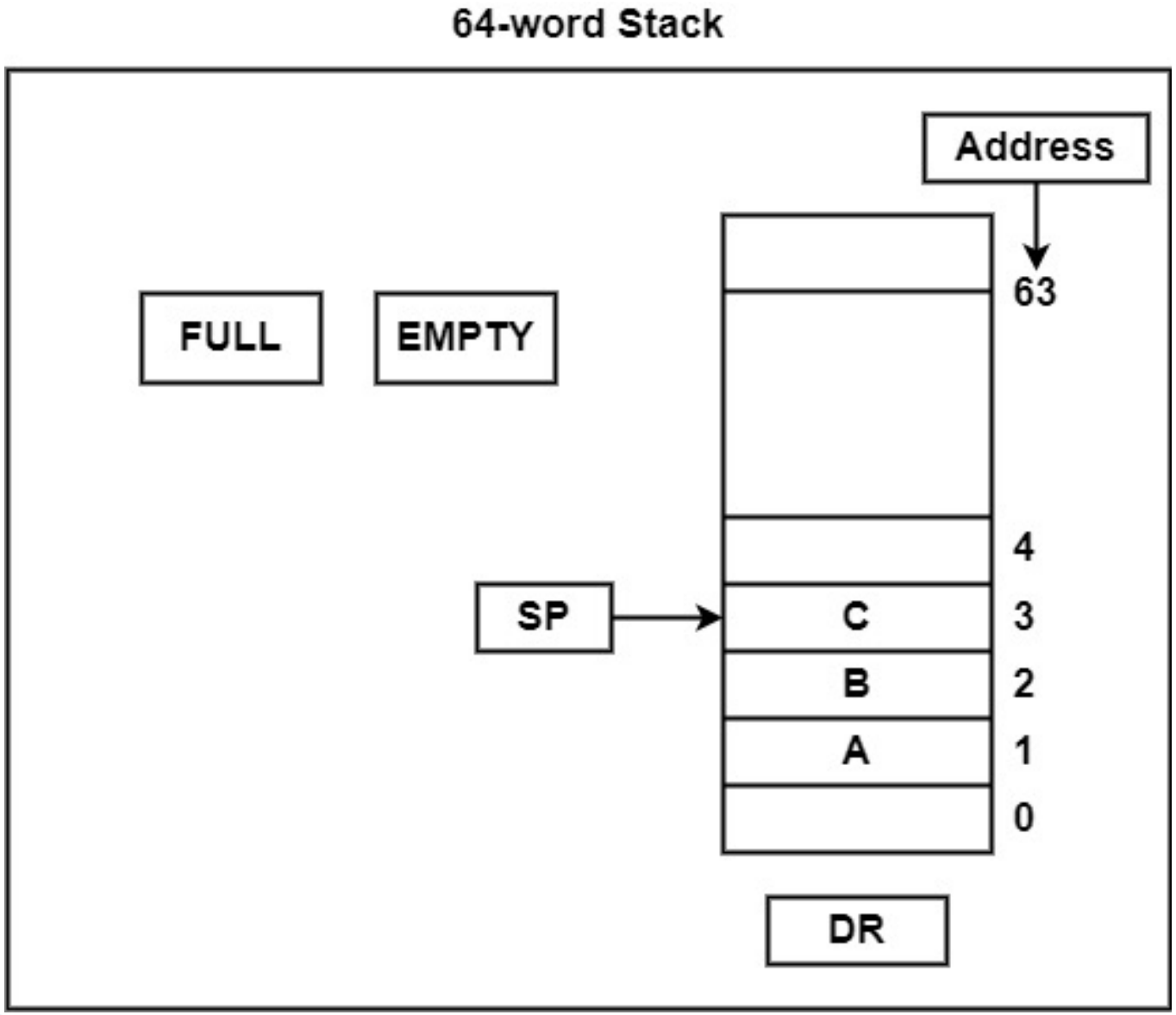
stack is also known as the Last In First Out **(LIFO)** list. It is the most important feature in the **cpu** . It saves data such that the element stored last is retrieved first.

A stack is a memory unit with an address register. This register influence the address for the stack, which is known as Stack Pointer (**SP**). The stack pointer continually influences the address of the element that is located at the top of the stack.

- It can insert an element into or delete an element from the stack.
- The insertion operation is known as **push** operation and
- the deletion operation is known as **pop** operation. In a computer stack, these operations are simulated by incrementing or decrementing the **SP** register.

Register Stack

- The stack can be arranged as a set of memory words or registers.
- Consider a 64-word register stack arranged as displayed in the figure. The stack pointer register includes a binary number, which is the address of the element present at the top of the stack. The three-element A, B, and C are located in the stack.
- The element C is at the top of the stack and the stack pointer holds the address of C that is 3. The top element is popped from the stack through reading memory word at address 3 and decrementing the stack pointer 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.


64-word Stack
Address
63
FULL
EMPTY
4
SP
C
3
B
2
A
1
0
DR

- The stack pointer includes 6 bits, because $2^6 = 64$, and the 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 push operation is executed as follows :

SP←SP + 1	It can increment stack pointer
K[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

- The stack pointer is incremented by 1 and the address of the next higher word is saved in the SP.
- The word from DR is inserted into the stack using the memory write operation.
- The first element is saved stack pointer is at 0, then the stack is full and ‘FULL’ is set to 1. This is the condition when the SP was in location 63 and after incrementing SP, the final element at address 1 and the final element is saved at address 0. If the is saved at address 0.
- During an element is saved at address 0, there are no more empty registers in the stack. The stack is full and the ‘EMTY’ is set to 0.
- A new element is deleted from the stack if the stack is not empty (if EMTY = 0).

The pop operation includes the following sequence of micro-operations :

DR←K[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

The top element from the stack is read and transfer to DR and thus the stack pointer is decremented. If the stack pointer reaches 0, then the stack is empty and 'EMTY' is set to 1. This is the condition when the element in location 1 is read out and the SP is decremented by 1