Chapter 6

Introduction to Central Processing Unit (CPU)

Topics to be covered:

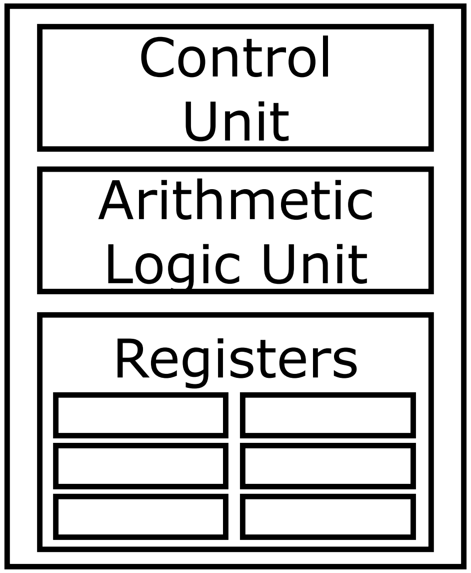
- Introduction to CPU
- Organization of CPU registers
    - Single Register Organization
    - General Register Organization
    - Stack Organization
- CISC and RISC

Central Processing Unit (CPU)

- The part of the computer that performs the bulk of data processing operations is called the central processing unit (CPU)
- The **central processing unit (CPU)** of a computer is the main unit that dictates the rest of the computer organization
- The CPU is made up of three major parts:
  - Register set
  - ALU
  - Control units

Central Processing Unit (CPU)

- 1. **Register set**: Stores intermediate data during the execution of instructions;
- 2. **Arithmetic logic unit (ALU)**: Performs the required micro-operations for executing the instructions;
- 3. **Control unit**: supervises the transfer of information among the registers and instructs the ALU as to which operation to perform by generating control signals.



Registers

- In Basic Computer, there is only one general purpose register, the **Accumulator** (AC)
- In modern CPUs, **there are many general purpose registers.**
- It is advantageous to have many registers:
  - Transfer between registers within the processor are relatively fast
  - Going “off the processor” to access memory is much slower
    - Because memory access is the most time consuming operation in a computer

Registers cont…

- There are three types of register organization
- These internal organization of registers (The three most common CPU organizations) are:
    - Single Accumulator Organization
    - General Register organization
    - Stack organization

6.2. General Register Organization

- CPU must have some working space (fast access and close to CPU)
- This space is efficiently used to store intermediate values
- Intermediate data are needed to be stored like Pointers, counters, return address, temp results, and partial products.
- Cannot save them in main memory because their access is time consuming.
- It is more efficient and faster to be stored inside processor.
- So the solution is **designing multiple registers inside processor and connects them through a common bus**.

6.2. General Register Organization

- Bus organization for 7 CPU registers:
  - These 7 registers are connected through the a common bus system
    - **2 MUX**: select one of 7 register or external data input by **SELA** and **SELB**
    - **BUS A and BUS B** : form the inputs to a common ALU
    - **ALU** : **OPR** determine the arithmetic or logic microoperation
      - The result of the microoperation is available for external data output and also goes into the inputs of all registers
    - **3 X 8 Decoder**: select the register (by **SELD**) that receives the information from ALU

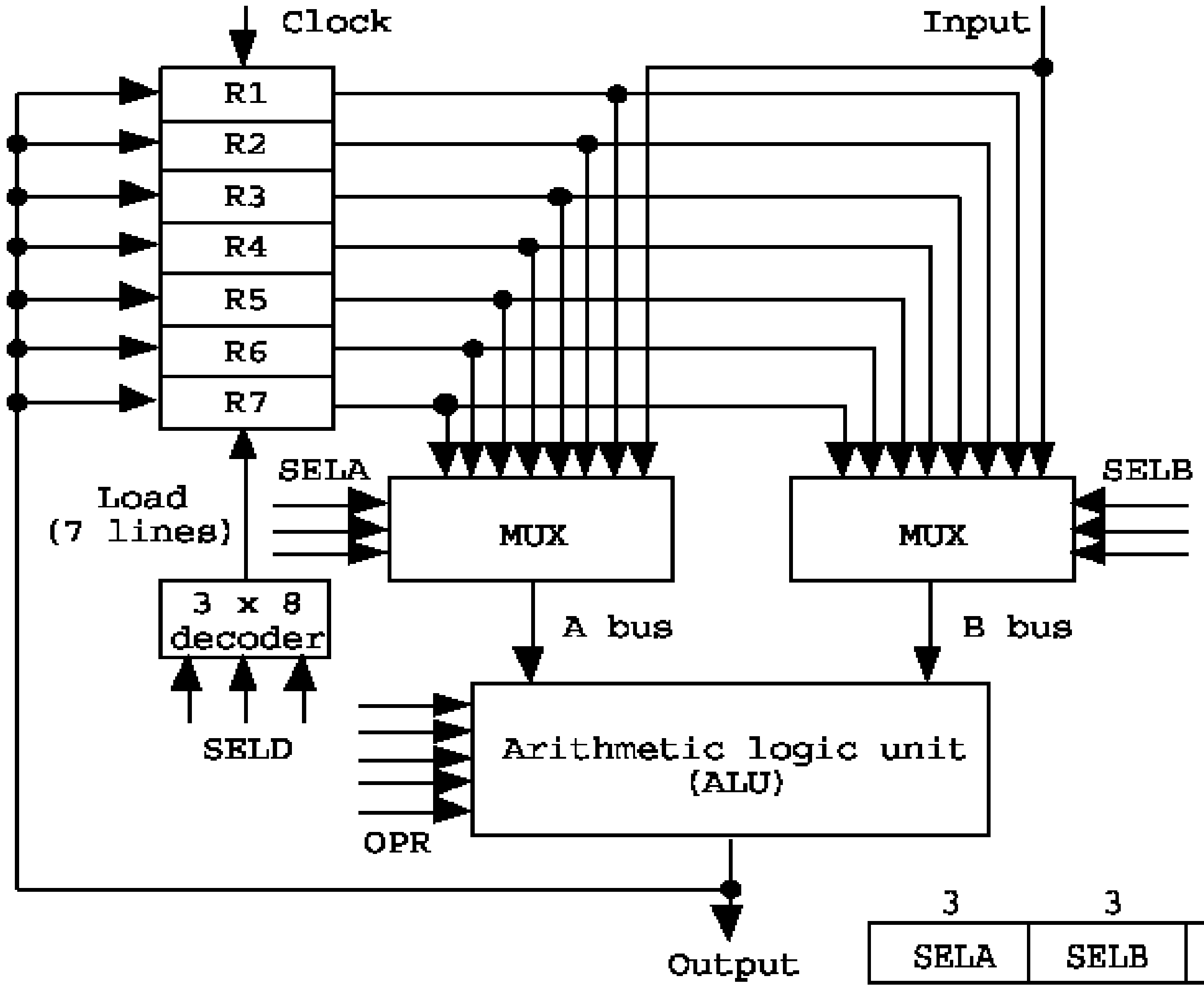
6.2. General Register Organization

- An operation is selected by the ALU **operation selector** (OPR).
- The result of a microoperation is directed to a destination register selected by a **decoder** (SELD).
- **Control wor**d: The 14 binary selection inputs (3 bits for SELA, 3 for SELB, 3 for SELD, and 5 for OPR)

3	3	3	5
SELA	SELB	SELD	OPR

- These four control signals are generated in **control unit** in start of each clock cycle ensuring operands are selected beside correct ALU operation and result is chosen in one clock cycle only.

6.2. General Register Organization



a) Block Diagram

3	3	3	5
SELA	SELB	SELD	OPR

b) Control Word

6.2. General Register Organization

- The control unit that operates the CPU bus system directs the information flow through the registers and ALU by selecting the various components in the system
- For example, to perform the operation

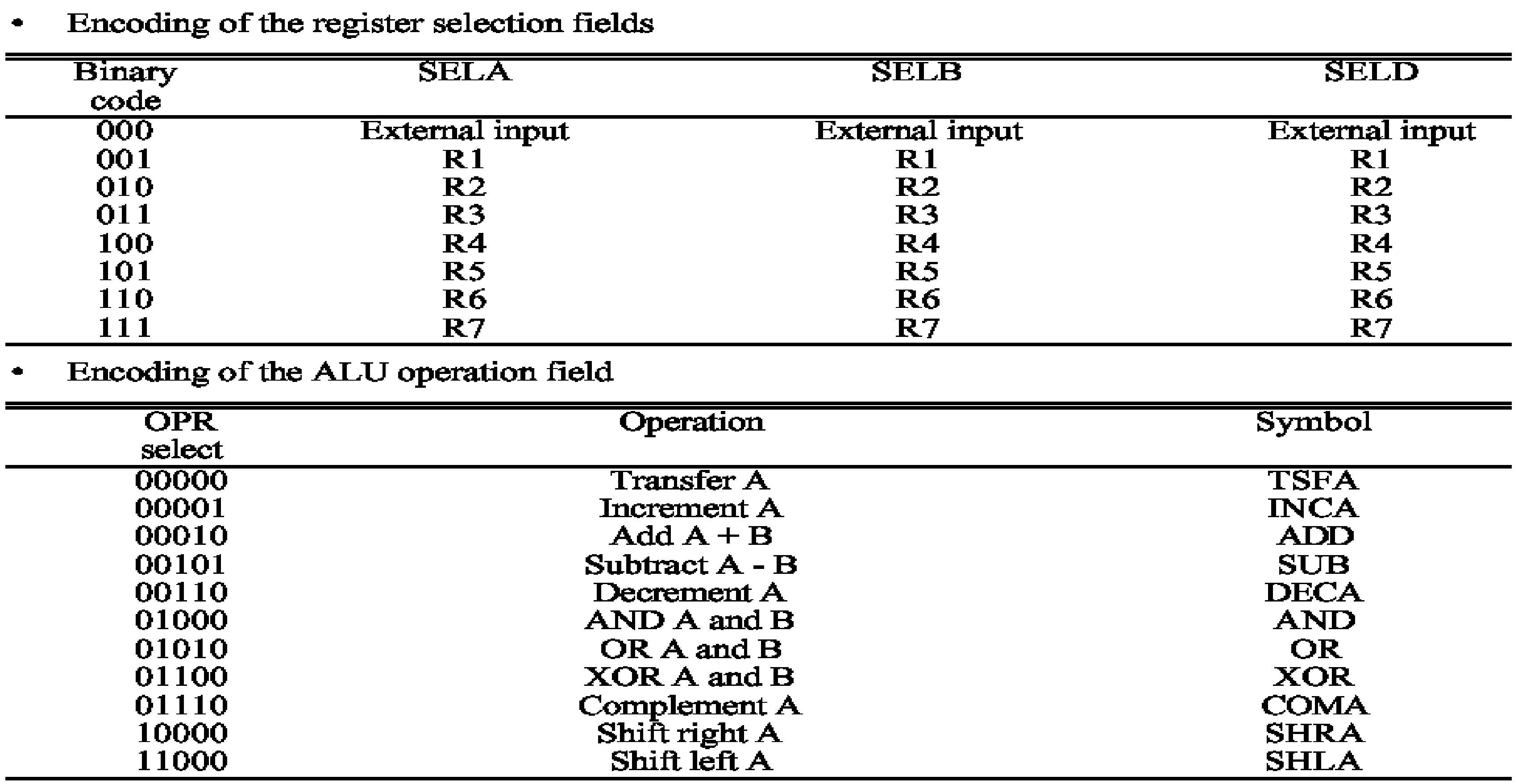
  $R1 \leftarrow R2 + R3$

  - The control unit must provide binary selection variables to the following selector inputs:

3	3	3	5
SELA	SELB	SELD	OPR

◆Binary selector input

1. MUX A selector **(SELA)** : to place the content of R2 into BUS A
2. MUX B selector **(SELB)** : to place the content of R3 into BUS B
3. ALU operation selector **(OPR)** : to provide the arithmetic addition R2 + R3
4. Decoder selector **(SELD)** : to transfer the content of the output bus into R1



- Encoding of the register selection fields

Binary code	SELA	SELB	SELD
000	External input	External input	External input
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 the ALU operation field

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	AND 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

- Encoding of Register Selection Fields:

»**SELA** or **SELB** = 000 (External **Input**) : MUX selects the external data

»**SELD** = 000 (**None**) : no destination register is selected but the contents of the output bus are available in the external output

6.2. General Register Organization

Example:

1. **Micro-operation:** $R1 \leftarrow R2 + R3$
2. **Control word**

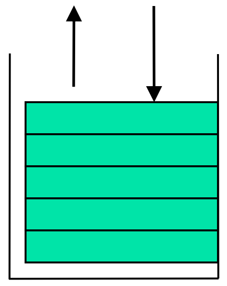
Field:	SELA	SELB	SELD	OPR
Symbol:	R2	R3	R1	Add
Control word:	*010*	*011*	*001*	*00010*

Q: let the content of R2=0001 and R3=0010, Please specify which register is selected by the decoder and what is the value going to be stored in the target register using the above given instruction in the control word.

Ans: R1, 0011

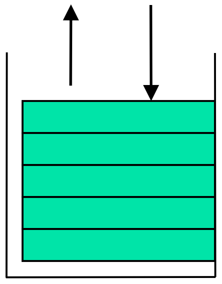
6.3 Stack Organization

- A useful feature that is included in the CPU of most computers is a stack or last-in, first-out(LIFO) list
- **Stac**k: A storage device that stores information in such a manner that the item stored last is the first item retrieved.
- The operation of a stack can be compared to a stack of trays. The last tray placed on top of the stack is the first to be taken off.



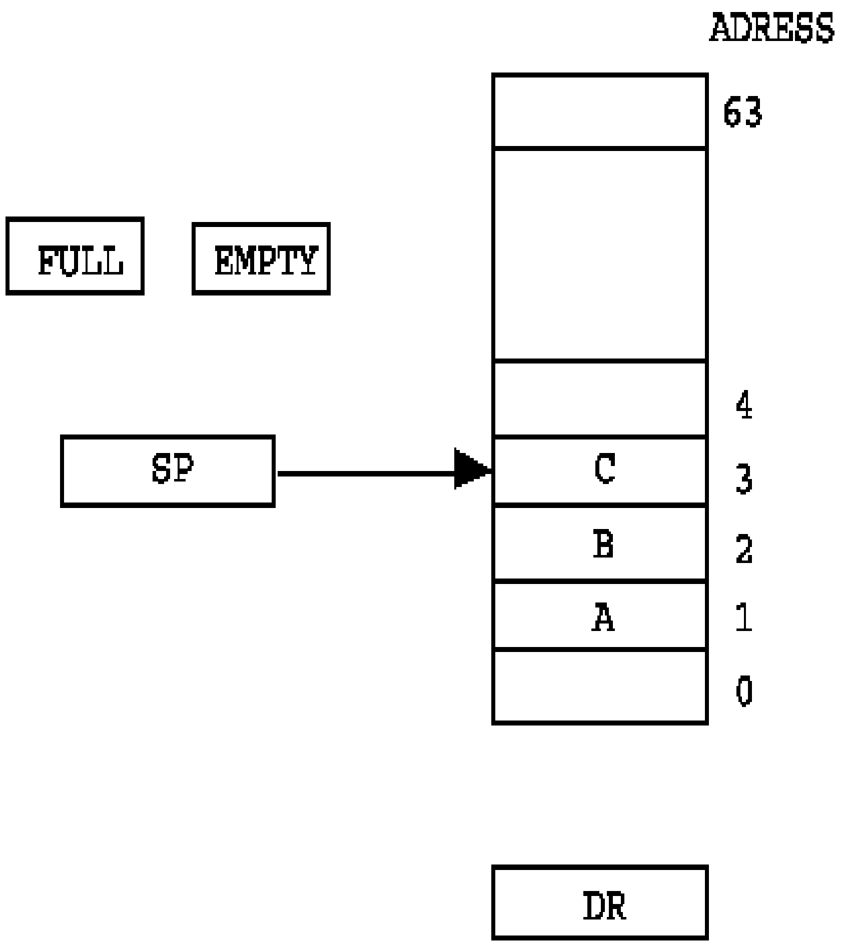
6.3 Stack Organization

- **Stack pointer (SP)**: A register that holds the address of the top item in the stack.
- **SP** always points at the top item in the stack
- There are exactly ***two operations*** that can be performed on a stack.
  - **Push (Push-down)**: it is the result of pushing a new item on top
    - Operation to insert an item into the stack.
    - It adds a new item to the top of the stack
  - **Pop (Pop-up)**: it is the result of removing an item from the top
    - Operation to retrieve an item from the stack.
    - It removes an item from the top of the stack.



Register Stack

- A stack can be organized as a collection of a finite number of registers.
- In a 64-word stack, the stack pointer contains 6 bits.
- The one-bit register
  - *FULL is set to 1 when the stack is full;*
  - *EMPTY register is 1 when the stack is empty.*
- The data register DR holds the data to be written into or read from the stack.



The following are the micro-operations associated with the stack

- Three items are placed in the stack: A,B, and C in that order
- The ***push operation*** is implemented with the following sequence of microoperation.

$SP \leftarrow 0$, $EMPTY \leftarrow 1$, $FULL \leftarrow 0$ (**Initialization)**

$SP \leftarrow SP + 1$ *(increment the pointer)*

$M[SP] \leftarrow DR$ *(Write item on top of the stack)*

If $(SP = 0)$ then $(FULL \leftarrow 1)$ *(Check if the stack is full)*

$EMPTY \leftarrow 0$ *(mark the stack not empty)*

- ***Note that SP becomes 0 after 63***

The following are the micro-operations associated with the stack

- A new item is deleted from the stack if the stack is not empty.
- The ***pop operation*** consists of the following sequence of micro-operations.

```
DR  ← M[SP]                          (Read item from the top of stack)
SP  ← SP - 1                         (decrement the stack pointer)
If(SP=0) then EMPTY ← 1)( check if the stack is empty)
FULL ← 0                       ( mark the stack not full)
```

Reverse Polish Notation (RPN)

- A stack organization is very effective for **evaluating arithmetic expressions**
- The common mathematical method of writing arithmetic expressions imposes difficulties when evaluated by a computer
- The common arithmetic expressions are written in infix notation, with each operator written between the operands.
- **Reverse polish notatio**n : is a **postfix notation** (places operators after operands)
- **Example : Consider the following expression:**
  - A + B   Infix notation
  - +AB   Prefix or polish notation
  - AB+   Postfix or Reverse Polish notation

Reverse Polish Notation (RPN)

- The reverse Polish notation is in a form suitable for stack manipulation.
- The following expressions are written in reverse polish notation as:
  - A * B + C * D $\rightarrow$ (AB *)+(CD *) $\rightarrow$ AB * CD * +
  - (3 * 4) + (5 * 6) $\rightarrow$ 34 * 56 * +

Evaluation procedure:

1. Scan the expression from left to right.
2. When an **operator** is reached, perform the operation with the two operands found on the left side of the operator.
3. Replace the two operands and the operator by the result obtained from the operation.

Reverse Polish Notation (RPN)

- ***Example: consider the following expression***
  - Infix ***(3 * 4) + (5 * 6) = 42***
  - Reverse polish notation: ***3 4 * 5 6 * +***

stack evaluation:

- ***Get value***
- ***If value is data: push data***
- ***Else if value is operation: pop, pop evaluate and push.***

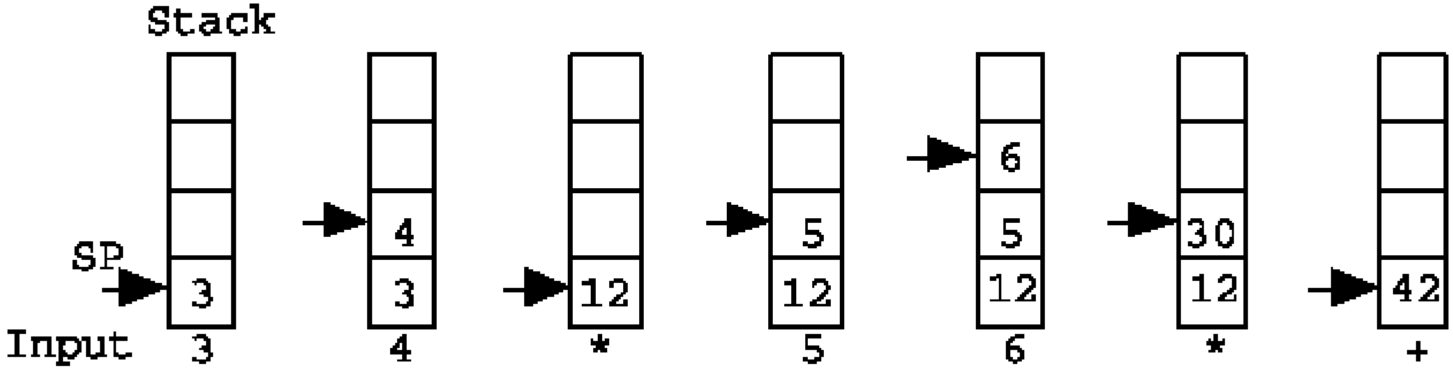
12 5 6 * +

12 30 +

42

Reverse Polish Notation (RPN)

- Reverse polish notation : ***3 4 * 5 6 * +***
- ***Consider the stack operation below:***
  - First the number 3 is pushed to the stack then the number 4
  - The next symbol is the multiplication operator *
    - This causes a multiplication of the two topmost items in the stack
  - The stack is then popped and the product is placed on top of the stack, replacing the two original operands.



6.4 Complex Instruction Set Computers: CISC

- A computer with **large number of instructions** is called complex instruction set computer or CISC.
- Complex instruction set computer is mostly used in scientific computing applications requiring lots of floating point arithmetic.

CISC Characteristics

- A large number of instructions - typically from 100 to 250 instructions.
- Some instructions that perform specialized tasks and are used infrequently.
- A large variety of addressing modes - typically 5 to 20 different modes.
- Variable-length instruction formats
- Instructions that manipulate operands in memory.

Reduced Instruction Set Computers: RISC

- A computer with **few instructions and simple construction** is called reduced instruction set computer.
- RISC architecture is simple and efficient.

The major characteristics of RISC architecture are:

- – Relatively few instructions
- – Relatively few addressing modes
- – Memory access limited to load and store instructions
- – All operations are done within the registers of the CPU
- – Fixed-length and easily-decoded instruction format.