CSE211

Computer Organization and Design

- ***General Register Organization***
- ***Stack Organization***
- ***Instruction Formats***
- ***Addressing Modes***

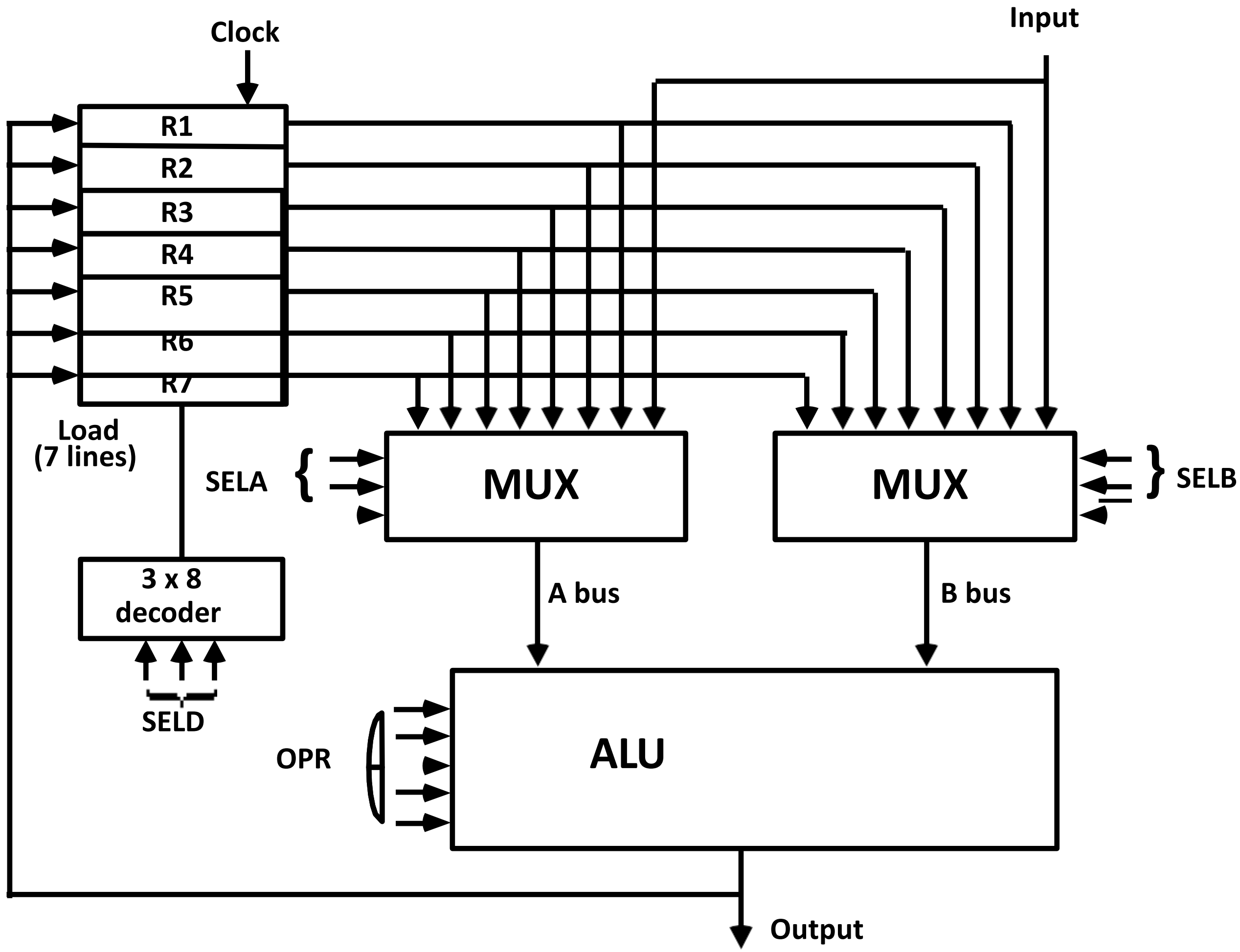
Major Components of CPU

- **Storage Components**
  - **Registers**
  - **Flags**
- **Execution (Processing) Components**
  - **Arithmetic Logic Unit(ALU)**
    - **Arithmetic calculations, Logical computations, Shifts/Rotates**
- **Transfer Components**
  - **Bus**
- **Control Components**
  - **Control Unit**

Processor Organization

- **In general, most processors are organized in one of 3 ways**
  - **Single register (Accumulator) organization**
    - **Basic Computer is a good example**
    - **Accumulator is the only general purpose register**
  - **General register organization**
    - **Used by most modern computer processors**
    - **Any of the registers can be used as the source or destination for computer operations**
  - **Stack organization**
    - **All operations are done using the hardware stack**
    - **For example, an OR instruction will pop the two top elements from the stack, do a logical OR on them, and push the result on the stack**

General Register Organization



EXAMPLE:

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

1. **SEL A** : **001** -To place the contents of R1 into bus A.
2. **SEL B** : **010** - to place the contents of R2 into bus B
3. **SEL OPR** : **10010** – to perform the arithmetic addition A+B
4. **SEL REG or SEL D** : **011** – to place the result available on output bus in R3.

Register and multiplexer input selection code

Binary code	SELA	SELB	SELD or SELREG
000	Input	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

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

$$R1 \leftarrow R2 - R3$$

Field:	SELA	SELB	SELD	OPR
Symbol:	R2	R3	R1	SUB
Control word:	010	011	001	00101

TABLE 8-3 Examples of Microoperations for the CPU

	Symbolic Designation				
Microoperation	SELA	SELB	SELD	OPR	Control Word
$R1 \leftarrow R2 - R3$	R2	R3	R1	SUB	010 011 001 00101
$R4 \leftarrow R4 \vee R5$	R4	R5	R4	OR	100 101 100 01010
$R6 \leftarrow R6 + 1$	R6	—	R6	INCA	110 000 110 00001
$R7 \leftarrow R1$	R1	—	R7	TSFA	001 000 111 00000
Output $\leftarrow R2$	R2	—	None	TSFA	010 000 000 00000
Output $\leftarrow$ Input	Input	—	None	TSFA	000 000 000 00000
$R4 \leftarrow$ shl $R4$	R4	—	R4	SHLA	100 000 100 11000
$R5 \leftarrow 0$	R5	R5	R5	XOR	101 101 101 01100

Stack Organization

Stack

Very useful feature for nested subroutines, nested interrupt services

Also efficient for arithmetic expression evaluation

Storage which can be accessed in LIFO

Pointer: SP

Only PUSH and POP operations are applicable

Stack Organization

Register Stack Organization

Memory Stack Organization

Register Stack Organization

- The computers which use Stack-based CPU Organization are based on a data structure called **stack**.
- The stack is a list of data words.
- It uses **Last In First Out (LIFO)** access method which is the most popular access method in most of the CPU.
- A register is used to store the address of the topmost element of the stack which is known as **Stack pointer (SP)**.
- In this organisation, ALU operations are performed on stack data.
- It means both the operands are always required on the stack. After manipulation, the result is placed in the stack.

In a 64-word stack, the stack pointer contains 6 bits because $2^6 = 64$. since SP has only six bits, it cannot exceed a number grater than 63(111111 in binary). When 63 is incremented by 1, the result is 0 since 111111 + 1 =1000000 in binary, but SP can accommodate only the six least significant bits. Similarly, when 000000 is decremented by 1, the result is 111111. The one bit register Full is set to 1 when the stack is full, and the one-bit register EMTY is set to 1 when the stack is empty of items. DR is the data register that holds the binary data to be written in to or read out of the stack.

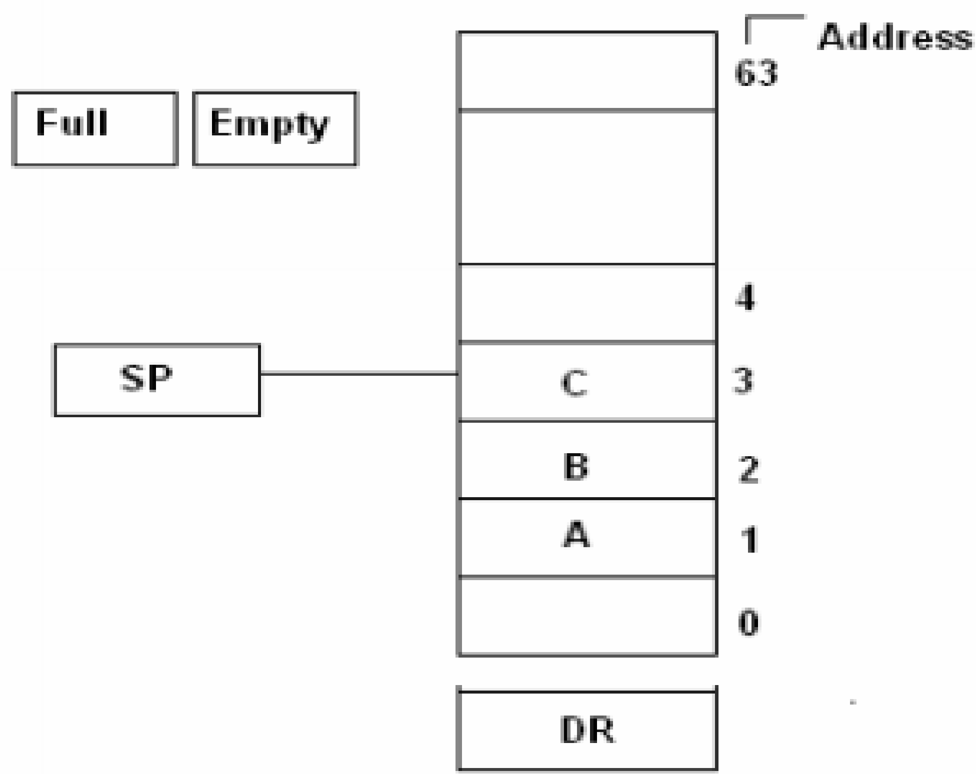
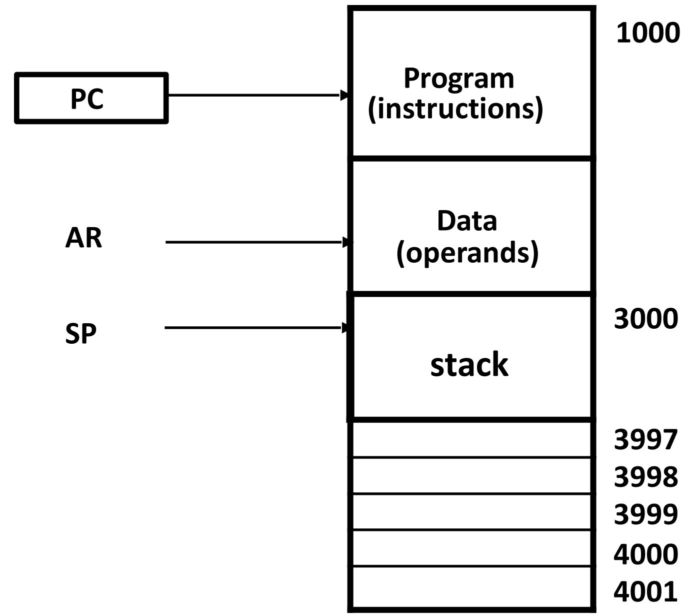


Figure :3 Block Diagram of a 64-word stack

Memory Stack Organization

Memory with Program, Data, and Stack Segments



- A portion of memory is used as a stack with a processor register as a stack pointer

- PUSH: SP ← SP - 1
  M[SP] ← DR

- POP: DR ← M[SP]
  SP ← SP + 1

- Most computers do not provide hardware to check stack overflow (full stack) or underflow (empty stack) <u>must be done in software</u>

Reverse Polish Notation

Stack is very effective in evaluating arithmetic expressions

- **Arithmetic Expressions:**

A * B + C * D

Polish Notation (Prefix) : Place operator before operand

Reverse Polish Notation (Postfix) : Place operator after operand

AB*CD*+

1. **(A*B)CD*+**
2. **(A*B) (C*D) +**
3. **(A*B) + (C*D)**

(A+B) * [C* (D+E)+ F] **AB+DE+C*F+***

Reverse Polish Notation

- **Arithmetic Expressions: A + B**

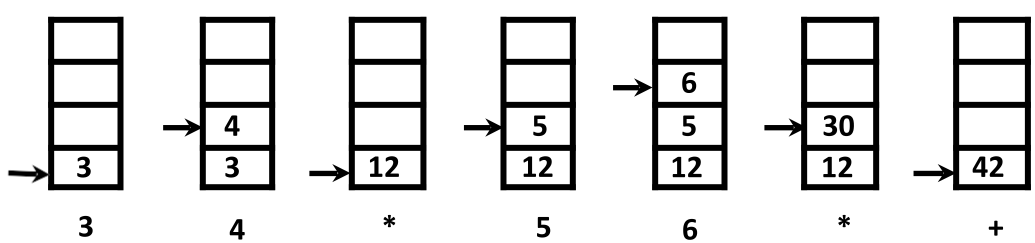
A + B	**Infix notation**
+ A B	**Prefix or Polish notation**
A B +	**Postfix or reverse Polish notation**

- The reverse Polish notation is very suitable for stack manipulation

- **Evaluation of Arithmetic Expressions**

Any arithmetic expression can be expressed in parenthesis-free Polish notation, including reverse Polish notation

(3 * 4) + (5 * 6) 3 **3 4 * 5 6 * +**



Instruction Format

- **Instruction Fields**

  OP-code field - specifies the operation to be performed

  Address field - designates memory address(es) or a processor register(s)

  Mode field - determines how the address field is to be interpreted (to get effective address or the operand)

- **The number of address fields in the instruction format depends on the internal organization of CPU**
- **The three most common CPU organizations:**

Single accumulator organization:

ADD	X	/* AC ← AC + M[X] */

General register organization:

ADD	R1, R2, R3	/* R1 ← R2 + R3 */
ADD	R1, R2	/* R1 ← R1 + R2 */
MOV	R1, R2	/* R1 ← R2 */
ADD	R1, X	/* R1 ← R1 + M[X] */

Stack organization:

PUSH	X	/* TOS ← M[X] */
ADD		

Three & Two Address Instruction

- **Three-Address Instructions**

Program to evaluate X = (A + B) * (C + D) :

```
ADD    R1, A, B     /*  R1 ← M[A] + M[B]     */
ADD    R2, C, D     /*  R2 ← M[C] + M[D]     */
MUL    X, R1, R2    /*  M[X] ← R1 * R2       */
```

- **Results in short programs**
- **Instruction becomes long (many bits)**

- **Two-Address Instructions**

Program to evaluate X = (A + B) * (C + D) :

```
MOV   R1, A     /* R1 ← M[A]        */
ADD   R1, B     /* R1 ← R1 + M[A]   */
MOV   R2, C     /* R2 ← M[C]        */
ADD   R2, D     /* R2 ← R2 + M[D]   */
MUL   R1, R2    /* R1 ← R1 * R2     */
MOV   X, R1     /* M[X] ← R1        */
```

-most common in commercial computer

One Address Instruction

- **One-Address Instructions**
  - **Use an implied AC register for all data manipulation**
  - **Program to evaluate X = (A + B) * (C + D) :**

LOAD	**A**	**/* $AC \leftarrow M[A]$ */**
ADD	**B**	**/* $AC \leftarrow AC + M[B]$ */**
STORE	**T**	**/* $M[T] \leftarrow AC$ */**
LOAD	**C**	**/* $AC \leftarrow M[C]$ */**
ADD	**D**	**/* $AC \leftarrow AC + M[D]$ */**
MUL	**T**	**/* $AC \leftarrow AC * M[T]$ */**
STORE	**X**	**/* $M[X] \leftarrow AC$ */**

Zero Address Instruction

- Zero-Address Instructions

  - **Can be found in a stack-organized computer**

  - **Program to evaluate X = (A + B) * (C + D) :**

PUSH	**A**	**/* TOS $\leftarrow$ A */**
PUSH	**B**	**/* TOS $\leftarrow$ B */**
ADD		**/* TOS $\leftarrow$ (A + B) */**
PUSH	**C**	**/* TOS $\leftarrow$ C */**
PUSH	**D**	**/* TOS $\leftarrow$ D */**
ADD		**/* TOS $\leftarrow$ (C + D) */**
MUL		**/* TOS $\leftarrow$ (C + D) * (A + B) */**
POP	**X**	**/* M[X] $\leftarrow$ TOS */**

Data Transfer and Manipulation

- Instruction set of different computers differ from each other mostly in way the operands are determined from the address and mode fields.

The basic set of operations available in a typical computer are :

Data Transfer Instructions

Data Manipulation Instruction :

perform arithmetic, logic and shift operation

Program Control Instructions

decision making capabilities, change the path taken by the program when executed in computer.

Data Transfer Instructions

Move data from one place in computer to another without changing the data content

Most common transfer : processor reg -memory, processor reg -I/O, between processor register themselves

- **Typical Data Transfer Instructions**

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

Commonly used data transfer operation:

Operation Name	Description
Move (Transfer)	Transfer word or block from source to destination
Store	Transfer word from processor to memory
Load (fetch)	Transfer word from memory to processor
Exchange	Swap contents of source and destination
Clear (reset)	Transfer word of 0s to destination
Set	Transfer word of 1s to destination
Push	Transfer word from source to top of stack
Pop	Transfer word from top of stack to destination

Data Transfer Instructions

Some assembly language conventions modify the mnemonic symbol to differentiate between the different addressing modes

- **Data Transfer Instructions with Different Addressing Modes**

Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	$AC \leftarrow M[ADR]$
Indirect address	LD @ADR	$AC \leftarrow M[M[ADR]]$
Relative address	LD $ADR	$AC \leftarrow M[PC + ADR]$
Immediate operand	LD #NBR	$AC \leftarrow NBR$
Index addressing	LD ADR(X)	$AC \leftarrow M[ADR + XR]$
Register	LD R1	$AC \leftarrow R1$
Register indirect	LD (R1)	$AC \leftarrow M[R1]$
Autoincrement	LD (R1)+	$AC \leftarrow M[R1], R1 \leftarrow R1 + 1$
Autodecrement	LD -(R1)	$R1 \leftarrow R1 - 1, AC \leftarrow M[R1]$

Data Maniplulation Instructions

These instruction performs operation on data and provide the computational capabilities for the computer

- **Three Basic Types:**

Arithmetic instructions
Logical and bit manipulation instructions
Shift instructions

Data Manipulation Instructions

Four basic arithmetic operations : + - * /

- **Arithmetic Instructions**

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with Carry	ADDC
Subtract with Borrow	SUBB
Negate(2's Complement)	NEG

Arithmatic:

Most machines provide the basic arithmatic operations like add, subtract, multiply, divide etc. These are invariably provided for signed integer (fixed-point) numbers. They are also available for floating point number.

The execution of an arithmatic operation may involve data transfer operation to provide the operands to the ALU input and to deliver the result of the ALU operation.

Commonly used data transfer operation:

	Operation Name		Description
	Add		Compute sum of two operands
	Subtract		Compute difference of two operands
	Multiply		Compute product of two operands
	Divide		Compute quotient of two operands
	Absolute		Replace operand by its absolute value
	Negate		Change sign of operand
	Increment		Add 1 to operand
	Decrement		Subtract 1 from operand

Data Manipulation Instructions

Logical Instructions perform binary operations on string of bits stored in registers

Useful for manipulating individual/ group of bits

Consider each bit separately

- **Logical and Bit Manipulation Instructions**

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

AND Clear selected bits

OR Set selected bits

XOR Complement selected bits

Logical:

Most machines also provide a variety of operations for manipulating individual bits of a word or other addressable units.

Most commonly available logical operations are:

	Operation Name		Description
	AND		Performs the logical operation AND bitwise
	OR		Performs the logical operation OR bitwise
	NOT		Performs the logical operation NOT bitwise
	Exclusive OR		Performs the specified logical operation Exculsive-OR bitwise
	Test		Test specified condition; set flag(s) based on outcome
	Compare		Make logical or arithmatic comparison Set flag(s) based on outcome
	Set Control Variables		Class of instructions to set controls for protection purposes, interrupt handling, timer control etc.
	Shift		Left (right) shift operand, introducing constant at end
	Rotate		Left (right) shift operation, with wraparound end

Data Manipulation Instructions

- **Shift Instructions**

Shift Instructions

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

Rotate left thru carry
ROLC

Input/Output :

Input/Output instructions are used to transfer data between input/output devices and memory/CPU register.

Commonly available I/O operations are:

	Operation Name		Description
	Input (Read)		Transfer data from specified I/O port or device to destination (e.g., main memory or processor register)
	Output (Write)		Transfer data from specified source to I/O port or device.
	Start I/O		Transfer instructions to I/O processor to initiate I/O operation.
	Test I/O		Transfer status information from I/O system to specified destination