UNIT-2

Computer Architecture → Catalogue of tools or attributes that are visible to the user such as instruction sets, number of bits used for data, addressing techniques, etc.

Computer Organisation → Defines the way system is structured so that all those catalogued tools can be used.

Computer Architecture	Computer Organisation
→ Way hardware components are connected together to form a computer system.	→ A structure and behaviour of a computer system as seen by the user.
→ Interface b/w hardware & software	→ Deal with the components of connection in a system.
→ Tells the functionalities of a system.	→ Tells us how exactly all the units in the system are arranged & interconnected.
→ A programmer can view archite. in terms of instructions, addressing modes and registers.	→ Organization express the realization of architecture.
→ While designing a computer system architecture is considered first.	→ Organization is done on the basis of architecture
→ Deals with high-level design issues.	→ Deals with low-level design issues.
→ Involves logic (Instruction sets, Addressing modes, Data types, Cache optimization)	→ Involves physical components (Circuit design, Adders, Signals, Peripherals)

~~Unit-2~~

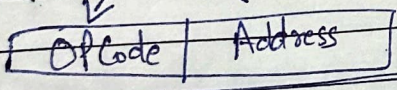
Computer Architecture :-

1.) Instruction set
2.) Instruction format
3.) Clock

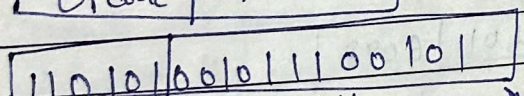
Computer Organisation :-

1.) ALU
2.) CU
3.) BUS, etc.

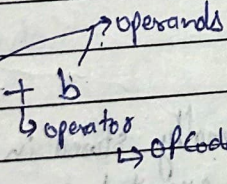
* Instructions → set of Number



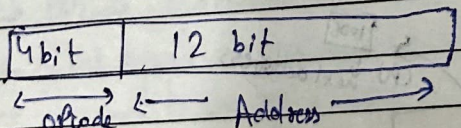
eg:-



eg:-



16 bit



$2^4 = 16$ — no. of instruction address

$2^{12} = 4096$

Mode	OPCode	Address

Tells that whether addressing mode is direct or indirect

Direct Mode - 0
Indirect Mode - 1

eg- of direct mode

0	1101	2698

address

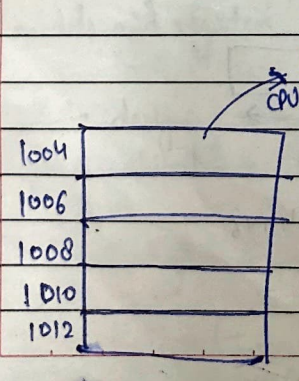
let address of 2698 → 6098

∴ indirect mode

1	1101	6098

Computer Registers :-

Computer instructions are normally stored in consecutive memory locations and are executed sequentially one at a time.



Computer Architecture :-

Instruction Codes in Computer Architecture :-

A computer instruction ~~code~~ is a binary code that specifies a sequence of micro operations for the computer. The instruction code with the data stored in memory. An instruction comprises groups called fields. These fields are:

- OPcodes (operation codes) → Specify how to perform a specific instruction
- Addresses → Specifies what to use — a register or an area, for a particular action. Contains operand's location

* Operands are precise computer instructions that show what information the computer requires to function.

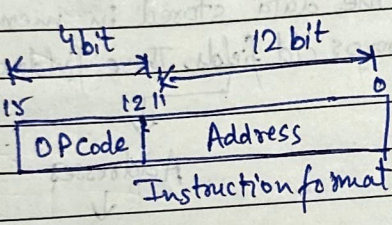
OPCode :-

→ Defines operations such as add, subtract, etc.
→ The Opcode must consist with n bits for a given 2^n distinct operation.
→ When the OPcode is decoded in the control unit (C.U) the computer issues signals to read an operand from memory and add the operand to processor register

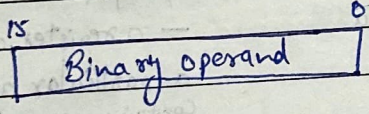
Stored Program Organisation :-

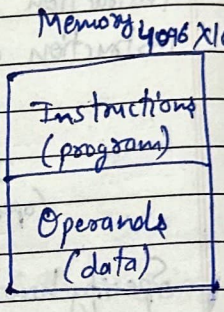
The simplest way to organise a computer is to have one processor register and an instruction code format with two

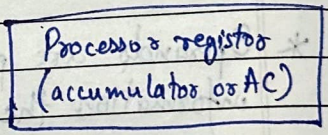
parts. The first part specifies the operation to be performed and the second specifies an address. The memory address tells the control where to find an operand in memory. This operand is read from the memory and used as the data to be operated on together with the data stored in the preprocessor register.



Instruction format







For a memory unit ~~to specify~~ with 4096 words we need 12 bits to specify any address, $2^{12} = 4096$.

Structure :-
If we store each instruction code in one 16-bit memory word, we've available four bit for the OP code to specify one out of, $2^4 = 16$, possible operations and 12-bits to specify the address of an operand.

Working :- The control reads a 16-bit instruction from the program portion of memory. It uses 12-bit address part of the instruction to read a 16-bit operand from the data portion of memory. It then executes the operation specified by the operation code.

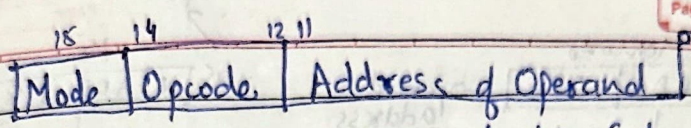


Fig: Structure of an Instruction Code

Mode → 0 (Direct), 1 (Indirect)
↳ Don't use address
↳ Use address

Direct Address:-

When we use the address bit of an instruction code not as an address but as the actual operand. When the second part of instruction specifies the address of an operand, the instruction is said to have direct address.

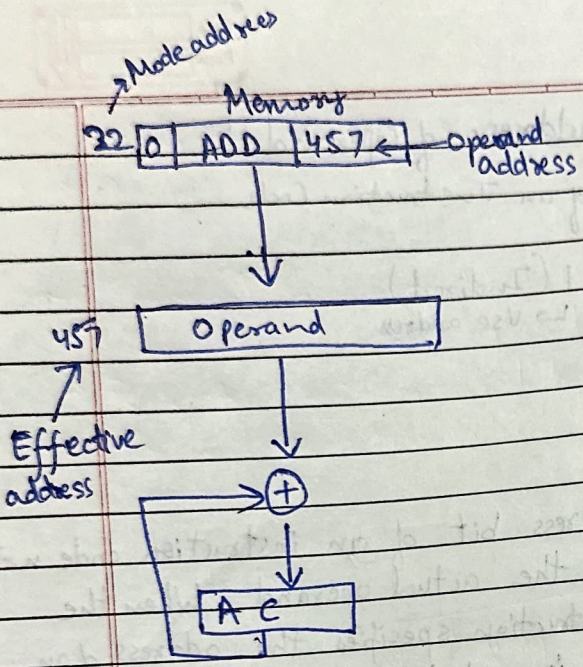
Indirect Address:-

Where the bits in the second part of the instruction designate an address of a memory word in which the address of the operand is found.

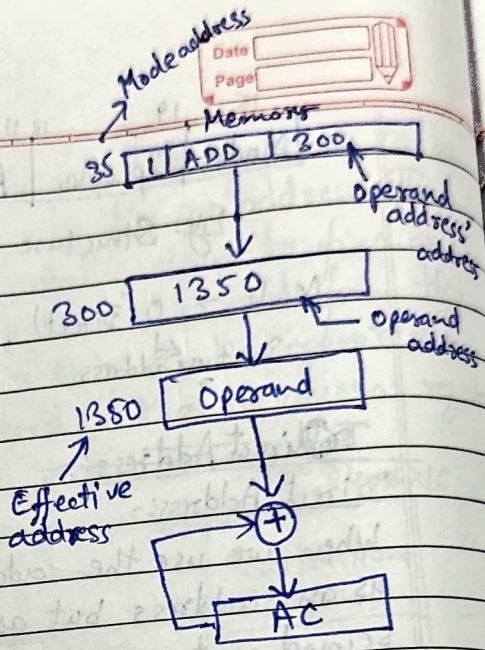
Effective Address :-

The address of the operand in computation-type instruction or the target address in a branch-type instruction.

* The memory "word" that holds the address of the operand in an indirect address instruction is used as a pointer to an array of data. The pointer could be placed in a processor register instead of memory as done in commercial computers.



Direct address



Indirect address

Computer Registers :-

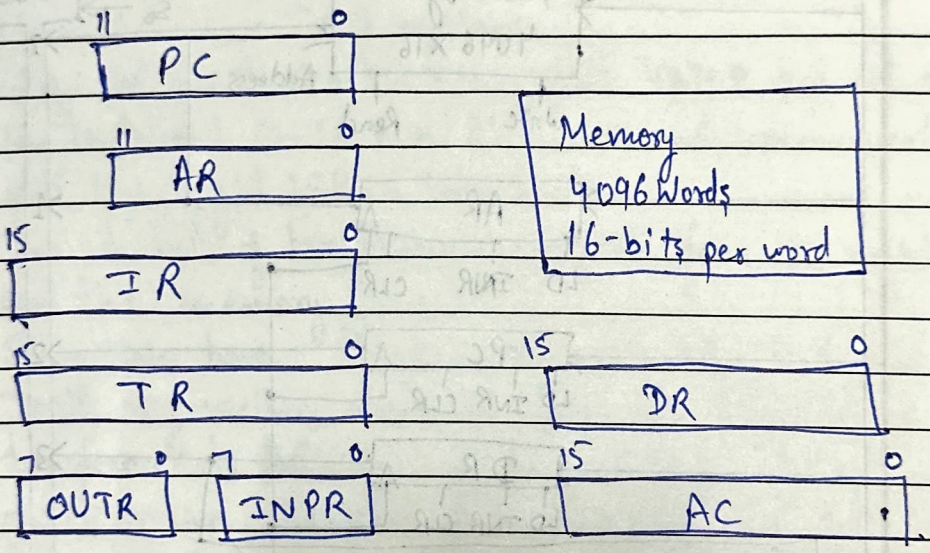
→ Computer instructions are normally stored in consecutive memory locations and are executed sequentially one at a time.

→ The instruction sequencing need a counter to calculate the address of the next instruction after execution of the current instruction is completed.

→ The computer needs processor of registers for manipulating data and a ~~registed~~ register for holding a memory address.

→ There are 8 types of registers :-

	Register Symbol	No. of bits	Name	Function
1.)	DR	16	Data Register	Holds memory operand
2.)	AR	12	Address Reg.	Holds address for memory
3.)	AC	16	Accumulator	Processor register
4.)	IR	16	Instruction Reg.	Holds instruction code
5.)	PC	12	Program Counter	Holds address of instruction
6.)	TR	16	Temporary Reg.	Holds temporary data.
7.)	INPR	8	Input Reg.	Holds input character
8.)	OUTR	8	Output Reg.	Holds output character



Program Counter (PC) :-

→ The AR has 12 bits which is width of memory address
→ The PC has 12 bits and it holds the address of next instruction
→ The PC goes through a counting sequence and causes the computer to read sequential instructions previously stored in memory. Instruction words are read and executed in sequence unless a branch instruction is encountered
→ A branch instruction calls for a transfer to a non-consecutive

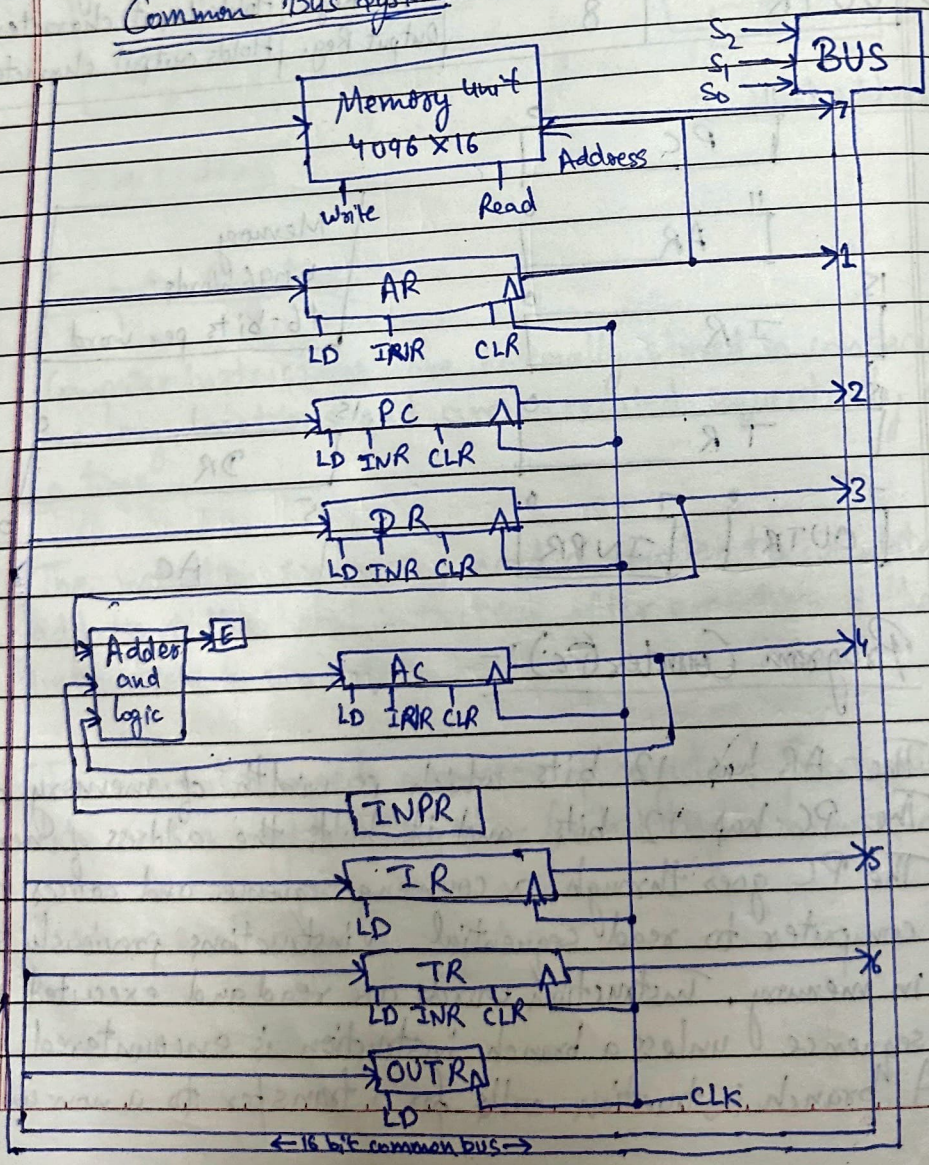
instruction in the program.

→ The address part of a branch instruction is transferred to PC to become the address of the next instruction.

→ To read an instruction, the content of PC is taken as the address for memory and memory-read cycle is initiated.

→ PC is then incremented by one, so it holds the address of the next instruction in sequence.

Common Bus System



→ The basic computer has eight registers, a memory unit, and a control unit.
→ A efficient way to transfer information in between these registers is to use common bus system.
→ The output of the seven registers (leaving INPR) and memory are connected to the common bus.
→ The selection lines are used to select register that can use bus.
→ The register whose LD (load) input is enabled receive the data from the bus.
→ Four registers DR, AC, IR and TR have 16 bits each.
→ AR and PC are 12 bit registers.
→ When AR and PC use bus the first 4 MSB are set to 0.
→ The INPR and OUTR have 8 bits each and communicate with the eight LSB in the bus.
→ The 16 bit of bus, recieve information from six registers and the memory.
→ 5 registers have three control inputs LD, INR and CLR.
→ INPR recieve character from the input device and transferred to AC.
→ OUTR recive character from the AC and delievers it to an output device.

Stack Organization :-

A stack is an ordered linear list in which all insertions and deletions are made at one end, called top. It uses Last In First Out (LIFO) access method which is the most popular access method in most of the CPUs.

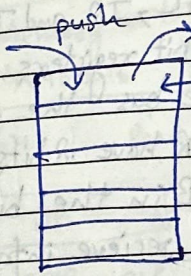
A register is used to store the address of the topmost

element of the stack which is k/a stack Pointer (SP) because its value always points at the top item in the stack.

There are two operations on stack.

→ Push operation: The operation of inserting an item onto a stack.

→ Pop operation: The operation of deleting an item onto a stack



Applications :-

→ Evaluation of mathematical expressions using Reverse Polish Notation.
→ To reverse a word. A given word is pushed to stack-letter by letter - and then popped out letters from the stack.
→ An undo mechanism in the text editor; this operation is accomplished by keeping all text changes in a stack.
→ Backtracking: this is a process when it is required to access the most recent data element in a series of elements.
→ Language processing.
→ Space for parameters and local variables is created initially using a stack.

→ The compiler's syntax check for matching braces is implemented by using stack.

There are two types of stack organization which are used in the computer hardware:

1) Register stack 2) Memory stack

1) Register Stack :-

It is built using register.

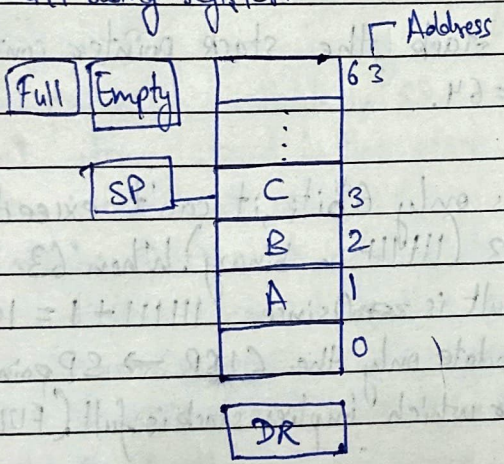


Fig: A 64 word register stack

→ We require 64 registers to make a register stack. The no. 0,1,2,3...63 denotes the address of different registers.

→ ~~SP is a pointer which points~~

→ SP is a stack pointer register which contains binary number whose value is equal to the address of the word that is currently at the top.

→ The two more registers called FULL and EMPTY are used. These are made up of flip-flops. It indicates whether the

stack is full or not.

- If FULL=1, then EMPTY=0 → stack is full
- If FULL=0, then EMPTY=1 → stack is empty.

→ DR is the data register through which data is transferred to and from the stack.

* Zero address instructions are used in register stack organization i.e. the address that does not contain the address of the operands.

In 64-word stack, the stack pointer contains 6 bits because $2^6 = 64$.

Since, SP has only 6 bits, it can't exceed a number greater than 63 (111111 in binary). When 63 is increamented by 1 the result is 0 since 111111 + 1 = 1000000, but SP can accomodate only the 6 LSB → SP points to 000000 address register which implies stack is full (FULL=1, EMPTY=0)

~~The PUSH operation is implemented with the following sequence of microoperations:~~

When 000000 is decreamented by 1, the result is 111111 → stack empty (FULL=0, EMPTY=1).

Operations:

(1) PUSH → Initially, SP is cleared to 0, EMPTY is set to 1 & FULL=0. Sequence of microoperations are as follows:

$SP \leftarrow SP + 1$	Increament stack pointer
$M[SP] \leftarrow DR$	Write item on top of stack
If $(SP = 0)$ the $(FULL \leftarrow 1)$	Check if stack is full
$EMTY \leftarrow 0$	Mark the stack is not empty

→ The stack pointer is increamented so that it points to the address of the next higher word.

→ A memory write operation inserts the word from DR into the top of the stack.

- SP holds the address of the top of the stack and that M[SP] denotes the memory word specified by the address presently available in SP.

→ The first item stored in the stack is at address 1. The last item is stored at address 0.

→ If SP reaches 0 → stack is full of items, so full is set to 1

- This condition is reached if the top item prior to the last push was in location 63, after increamenting SP, the last item is stored in location 0. Once an item is stored in location 0, there is no more empty registers in the stack.

→ If an item is written in the stack, obviously the stack can't be empty, so EMTY is cleared to 0.

(ii) POP → A new item is deleted from the stack if the stack is not empty (if EMTY = 0). The POP operation consist of following sequence of microoperations:

$DR \leftarrow M[SP]$	Read item from top of the stack
$SP \leftarrow SP - 1$	Decreament stack pointer
If $(SP = 0)$ then $(EMTY \leftarrow 1)$	Check if stack is empty.
$FULL \leftarrow 0$	Mark the stack not full.

→ The top item is read from the stack into DR.
→ The stack pointer is then ~~decrem~~ decremented.
→ If its value reaches zero, the stack is empty, so EMTY is set to 1. This condition is reached if the item read was in location 1.
→ Once this item is read out, SP is decremented and reaches the value 0.
→ If pop operation reads the item from location 0 and then SP is decremented, SP changes to 111111, which is equivalent to decimal 63.
• In this configuration, the word in address 0 receives the last ~~element~~ item in the stack.

* An erroneous operation will result if the stack is pushed when FULL=1 or popped when EMTY=1.

2) Memory Stack:-

It is logical part of memory allocated as stack. The logically partitioned part of RAM is used to implement stack.

* The implementation of stack in CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer (SP).

The RAM is divided into 3 logical parts:
(i) Program - The logical part of RAM where programs are stored.
(ii) Data - It is the logical part of the RAM where data (operands) are stored.

(ii) Stack - It is the part of RAM used to implement stack.

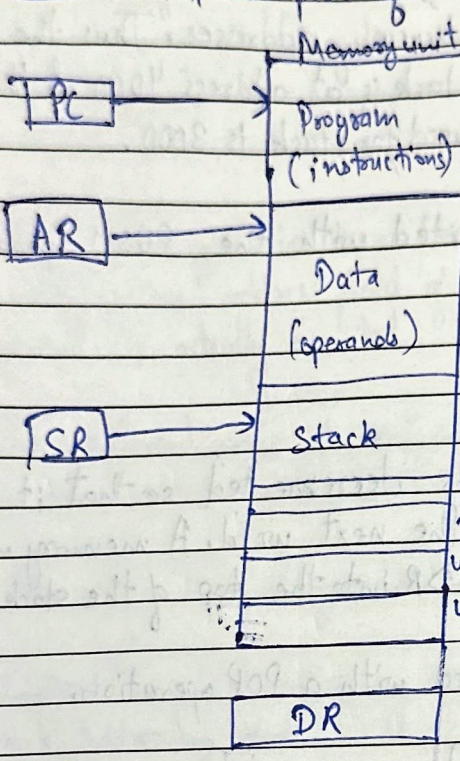


Fig: Memory Stack

→ The PC points at the address of the next instruction in the program.
→ AR points at an array of data.
→ SP points at the top of the stack.
→ These 3 registers are connected to a common address bus, and either one can provide an address for memory.
→ PC is used during the fetch phase to read an instruction. AR is used during the execute phase to read an operand. SP is used to push or pop items into or from the stack.
→ DR is used to store data. Data is pointed by address pointer or register (AR). Program Counter (PC) is used to point the program instruction.

As shown in the figure, the initial value of SP is 4001 and the stack grows with decreasing addresses. Thus the first item stored in the stack is at address 4000 & the last address that can be used for stack is 3000.

→ A new item is inserted with the PUSH operation.

$SP \leftarrow SP-1$
$M[SP] \leftarrow DR$

The stack pointer is decremented so that it points at the address of the next word. A memory write inserts the word from DR into the top of the stack.

→ A new item is deleted with a POP operation.

$DR \leftarrow M[SP]$
$SP \leftarrow SP+1$

The top item is read from the stack into DR. The stack pointer (SP) is then incremented to point at the next item in the stack.

* No provisions are available for stack limit checks

Addressing Modes :-

→ The term addressing modes refer to the way in which operand of an instruction is specified.

→ The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed.

→ The different ways of specifying the location of an operand in an instruction are cld as addressing modes.

There are 7 addressing modes:

1.) Implied/Implicit Addressing Mode.
2.) Stack Addressing Mode
3.) Immediate addressing mode.
4.) Direct addressing mode
5.) Indirect addressing mode
6.) Register Direct addressing mode.
7.) Register Indirect addressing mode.

1.) Implied addressing Mode :-

Definition of the instruction intself specify the operands implicitly.

eg:- → The instruction "Complement Accumulator" is an implied mode instruction (CMA)

→ In a stack organized computer, zero address instructions are implied mode instruction.