STORED PROGRAM ORGANIZATION.

one processor register: Accumulator.

Instruction code format: with 2 parts.

First part: operation to be performed: opcode

Second part: specifies address of operand.

→ Memory Address tells control where to find operand in memory.

→ This operand is read from memory. and used as the data to be operated on together with the data stored in the processor register.

2^{12} words in memory — 4096

→ If we store each instruction code in one 16 bit memory word. we have available 4 bits for the OPERATION CODE (opcode) to specify one out of 16 possible operations

12 bits to specify the address of an operand.

- the control reads a 16 bit instruction from the PROGRAM portion of memory. It uses the 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 opcode.

ACCUMULATOR: AC

→ If an operation in an instruction code doesnot need an operand from memory. Rest of the bits in instruction can be used for other purposes.

Examples of instructions that donot need an operand:

& INCREMENT AC

for these instructions 1 types of operations, the second part of the instruction code (bits '0' through '11') is not needed for specifying a MEMORY address. and can be used to specify other operations for the computer.

→ INDIRECT ADDRESS

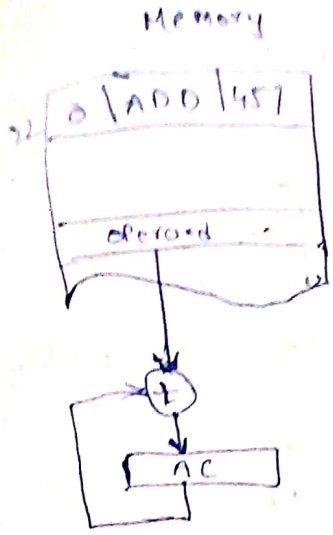
→ It is sometimes convineint to use the address bits of an instruction code, not as address, but as the actual operand. When the second part of an instruction code specifies an operand, the INSTRUCTION IS SAID TO HAVE AN IMMEDIATE OPERAND.

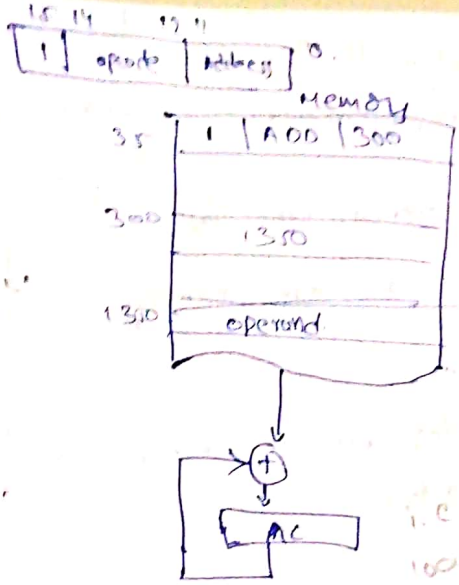
— when the second part specifies the address of an operand the INSTRUCTION IS SAID TO HAVE A DIRECT ADDRESS

— A THIRD possibility called 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.

→ one bit of the instruction code can be used to distinguish between a direct address & indirect address.

→ Consider INSTRUCTION CODE format shown in Fig (a) It consists of 3 bit opcode, a 12-bit address and an indirect address mode bit designated by I. The mode bit is '0' for direct & '1' for indirect address. it is placed in





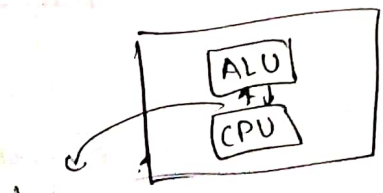
→ INDIRECT ADDRESS ... POSITION OF ADDRESS ... it can be loaded ... memory ... to

I bit is '0'. So the instruction is recognised as a direct address instruction. The opcode specifies ADD instruction and the address part is the binary equivalent of 457. The control finds the operand in memory at address 457 and add it to the content of accumulator.

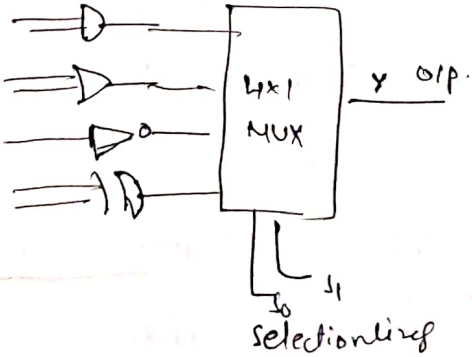
→ The INSTRUCTION in address 35 has mode bit I=1 Therefore it is recognized INDIRECT ADDRESS instruction. The address part is binary equivalent of contents at 300 The control goes to address 300 to find the address of the operand. So the address of the operand in this case is 300 → 1350. The operand found in address 1350 it is then added to content of accumulator.

EFFECTIVE ADDRESS

26 March

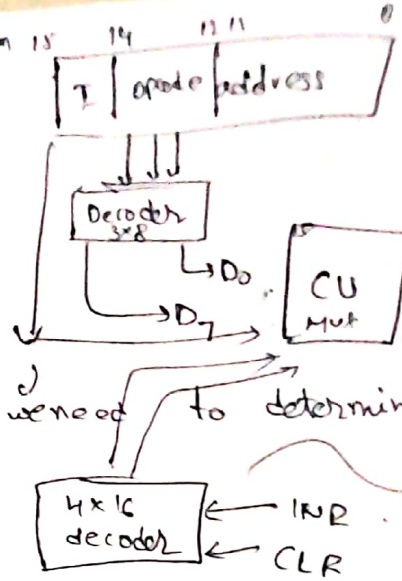


ALU



→ Timing & Control signal to ALU.

CU → Hard-wired → fast op. is
→ Micro programmed → slow problem is
- can't be changed easily



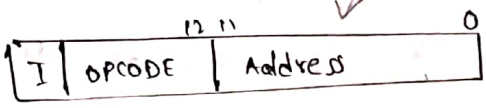
→ Register reference runs fast.
One more cycle than memory reference ??

we need to determine seperately

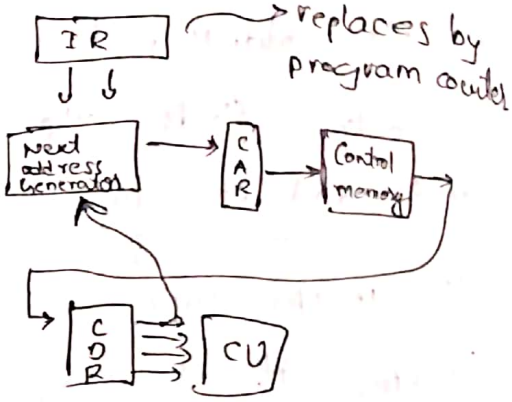
Sequence counter we can increase or decrease or clear.

⇒ Fundemental design of control unit with hardware.

MP ~ program which sends timing & seqn to ALU.



→ If we make control program modified easily control unit is changed.



→ Instruction decode, Instruction fetch.

pipelining ~ parallel execution of unrelated exp

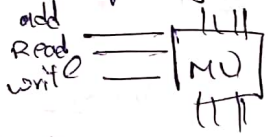
27 March

COMPUTER REGISTERS:

INSTRUCTION CODE
PROCESS REGISTER
MEMORY ADDRESS

- Computer instructions: stored in consecutive memory locations and are EXECUTED SEQUENTIALLY one at a time
- The control reads an instruction from a specific address in memory and executes it
- It then continues by reading the next instruction in sequence and executes it and so-on
- This type of instruction sequencing needs a COUNTER to calculate the address of the next instruction after execution of current instruction is completed

- It is also necessary to provide a register in the control unit for storing the INSTRUCTION CODE after it is read from memory.
- The computer needs PROCESSOR REGISTERS for manipulating data
- A register for holding the memory address
- These requirements dictate the REGISTER CONFIGURATION in figure.
- The registers are listed in Table 5.1 together with a brief of their function and the # of bits they contain.
- The memory unit has a CAPACITY of 4096 words and each word contains 16 bits. Twelve bits of an INSTRUCTION WORD are needed to specify the address of an OPERAND. This leaves 3 bits for operation part (opcode) of instruction and a bit to specify a direct or indirect address



- The DATA REGISTER (DR) holds the operand read from memory
- The ACCUMULATOR register is a general purpose processing register.
- The instruction read from memory is placed in the INSTRUCTION REGISTER (IR)
- The temporary register (TR) is used for holding temporary data during processors.
- The MEMORY ADDRESS REGISTER (MAR) has 12 bits. Since this is the width of a memory address.

→ Indirect is used when it is relatively known

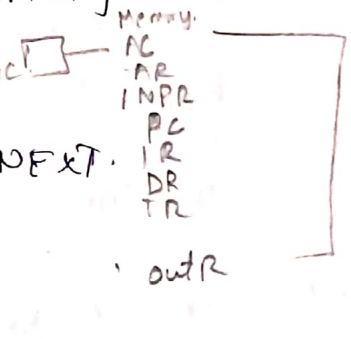
IR, TR, MAR, PC

- The PROGRAM COUNTER (PC) also has 12 bits and it holds address of the NEXT INSTRUCTION to be read from memory after the current instructions previously stored in memory

- Instruction words are read and executed in a sequence unless 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 PROGRAM COUNTER to become the address of next instruction.

- To read an instruction, the content of PC is taken as the address from the memory and a memory read cycle is initiated



- PC is incremented by one --- NEXT.

March 29 COMMON BUS SYSTEM

- The BASIC COMPUTER under DISCUSSION;
  Has 8 registers, a MEMORY unit, & a CONTROL UNIT
  - Paths to transfer information from one register to another and between Memory & registers
  - The number of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers
  - A more efficiat scheme for transfering information in a system with many register is to a

the connection of the registers and MEMORY of the BASIC computer to a common bus shown in figure

- the outputs of 7 registers and memory are connected to the COMMON BUS. The specific output that is selected for the bus lines at any given time is determined from the binary value of the selection variables S_2, S_1 & S_0. The number along each output shows the DECIMAL equivalent of the required binary selection. For example the number along the output of DR is '3'. The 16 bit outputs of DR are placed on the bus lines when $S_2S_1S_0 = 011$. Since this is the binary value of decimal '3'.

- The lines from common bus are connected to the inputs of each register and the data inputs of the memory.

- The particular register whose LD (Load) input is ENABLED receives the data from the bus during the next clock pulse transition

- The memory receives the contents of the bus when its "write input is activated". The memory places its 16 bit output onto the bus when the READ input is activated

- The memory places its 16 bit output onto the bus when the READ input is activated and $S_2S_1S_0 = 111$

→ 4 registers, DR, AC, IR & TR have 16 bits each. Two registers AR & PC have 12 bits each since they hold a memory address

→ when the contents of AR or PC are applied

to the 16 bit common bus, the 4 most significant bits are set to 0's??

→ When AR or PC receives information from the bus, only 12 least significant bits are transferred into the register.

→ The INPUT REGISTER, INPR and the output register OUTR have 8 bits each and communicate with the LEAST SIGNIFICANT BITS in the bus. INPR is connected to provide information from the bus. This is because INPR receives a character from an input device which is then transferred to AC. OUTR receives a character from AC and delievers it to an output device. There is no transfer from OUTR to any of the other registers.

- The 16 lines of the common bus receive information from 6 registers and the memory

- The bus lines are connected to the inputs of 6 registers and the memory

- 5 registers have 3 control inputs.

3 control inputs: LD (load), INR (increment) & CLR (clear)

MEMORY ADDRESS:

- The input data and output data of the memory are connected to the common bus, but the memory address is connected to AR. Therefore AR must always be used to specify a memory address.

By USING a SINGLE REGISTER [MAR] for address we ELIMINATE the need for an ADDRESS BUS that would have been needed otherwise.

- The 16 bits of AC come from an adder & logic circuit. This circuit has 3 sets of inputs. One set of 16-bit inputs come from the outputs of AC. These 16 bit inputs are used to implement register microoperations. Such as Complement AC & Shift AC

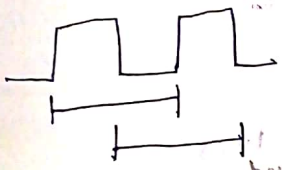
- Another set 16-bit inputs come from DATA REGISTER (DR). The inputs from Data register & AC are used for Arithmetic operations & logic microoperations such as ADD DR to AC. (or) AND DR to AC.

- The result of an addition is transferred to an accumulator and the end carry of the addition is transferred to flip-flop E (extended AC bit).

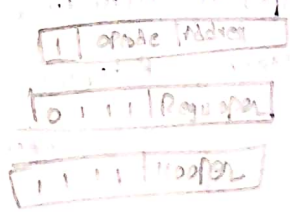
- The third set of 8-bit inputs come from the input register INPR.

Note: The content of any register can be applied on to the bus and an operation can be performed in the adder and logic circuit during the same clock cycle.

- The clock transition at the end of the cycle transfers the content of the bus into the designated destination register.



& the output of adder & logic circuit into the AC.



Computer Instructions: Instruction formats

- Instruction format: The basic computer has 3 instruction code formats. Fig 5-5 each format has 16-bits

- The opcode part of the instruction contains 3 bits & the meaning of the remaining 13 bits depends on the opcode encountered.

- A memory reference instruction uses 12 bits to specify an addressed one bit to specify the addressing mode I. I has the value '0' for direct address & '1' for indirect address.

- The Register reference instructions are recognized by the operation code 111 with a '0' in the leftmost bit of instruction.

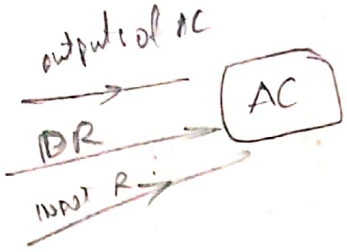
- A register reference instruction specifies an 'operation' on or a test of the AC register.

- An operand from memory is not needed, therefore the other 12 bits are used to specify an operation or test to be executed

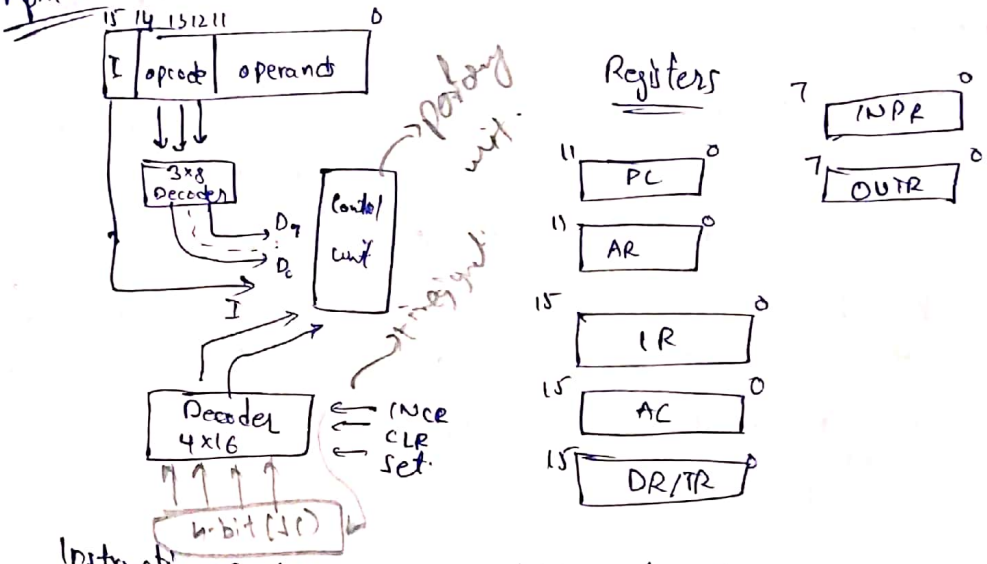
- Similarly, an I/O instruction does not need a reference to memory & is recognized by the opcode 111 & with leftmost bit as 1 of the instruction.

- The remaining 12 bits are used to specify the type of I/O operation or test performed.

- The type of instruction



April 2

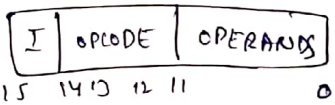


Instruction Cycle

1. Fetch
2. Decode
3. Effective address
4. Execute

Types of Instructions

1. Memory-Reference Instructions



2. Register-Reference Instruction

   12-14-bits — 111 — I-0

3. I-O Instruction

   12-14-bits - 111 — I-1

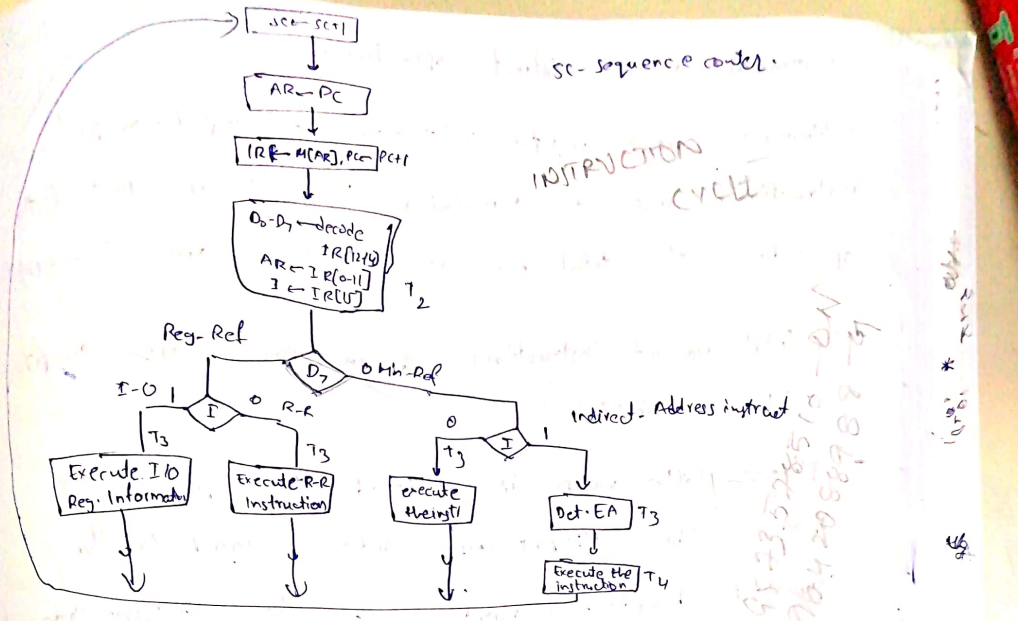
Fetch

T_0 : $AR \leftarrow PC$ T_1: $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

time 0

Decode : $I \leftarrow IR[15]$ $IR(0-11)$

T_2: D_7 : $D_i \leftarrow Decode\ [12-14]$, $AR \leftarrow IR(0-11)$, $I \leftarrow IR[15]$



SC - sequence counter.

INSTRUCTION CYCLE

April 3

→ we try to reduce number of instructions.

→ the type of instruction is recognised by the computer control. from the 4 bits in positions 12 through 15 of the instruction. If the three opcode bits in positions 12 through 14 are not equal to '111'. the instruction is a Memory reference type and the bit in position 15 is taken as the addressing mode I. If the 3 bit opcode is equal to 111, control then inspects bit in position 15. If this bit is '0' the INSTRUCTION is REGISTER REFERENCE type.

If this bit is 1, the INSTRUCTION is INPUT-OUTPUT type.

→ only 3 bits of the instruction are used for OPCODE

It may seen that the computer is restricted to a.

maximum of 8 distinct operations.

- However, since register reference and input-output instructions use the remaining 12 bits as part of the op-code, the total no of instructions can exceed eight.

→ Total No. of instructions chosen for basic computer = 25 --- Table 5.2.

→ Three letter WORD is utilized for represent the opcode. The HEXADECIMAL code is equal to the equivalent hexadecimal number of the binary code used for the instruction.

ADVANTAGE: Reduced the 16 bits of an instruction code to 4 digits.

→ Memory reference instruction: Address part of 12 bits is denoted by three x's. (3 hexadecimal digits corresponding to the 12-bit addresses).

→ The last bit of the instruction is designated by the symbol I when I=0, the last 4-bits of an instruction have a hex equivalent from '0' to '6'. (bcoz the last bit is '0') hex equivalent from 8 to 'E'.

INSTRUCTION SET COMPLETENESS:

- A computer should have a SET of instructions so that the user can construct MACHINE LANGUAGE PROGRAMS to evaluate any function that is known to be computable.

$f(n) = x^2 - 2 = 0$

April-4

- Modelling is done for predicting things.

Occam's Razor: Make the model simple, but not any simpler.

↳ principle of Passimony

Instruction set Completeness:

- A computer should have SET OF INSTRUCTIONS so that the user can construct MACHINE LANGUAGE programs to evaluate any function that is known to be COMPUTABLE

- the set of instructions is said to be complete if the computer includes a sufficient number of instructions in each of the following categories:

1. Arithmetic, logical, shift instructions.
2. Instructions for moving information to & from memory and processor register.

→ Memory reference instruction

3. Program Control instructions together with instructions that check status condition

→ In what order they are executed.

4. Input & Output instructions.

- The bulk of binary information in a digital computer (atleast stored program) is stored in memory, but all computations in processor register. Therefore the user must have the capability of moving information between these two units.

→ There is no optimal or best instruction set.

- Decision Making capabilities are an important aspect of DIGITAL COMPUTER's (DC's)

- Program control instructions eg: Branch & Branch --
- Input & output instructions. Needed for communication with (external word)

- The instructions listed in Table 5.2 constitute a MINIMUM SET that provides most of capabilities.

TIMING & CONTROL

- The timing for all registers in the BASIC COMPUTER is controlled by a master clock generator.
- The clock pulses are applied to all flip flops and registers in the system, including flip flops and registers in the Control unit.
- The clock pulses donot change the state of a register unless the register is enabled by a control signal.
- The control signals are generated in the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor register and microprog operations for the accumulator.
- There are 2 major types of control organization: Hardwired Control & Microprogrammed Control.

→ Hardwired Organization: CONTROL LOGIC is implemented with gates, flip-flops, decoders.

April 8

qubits → they can store 0's and 1's simultaneously are used in quantam computers

pg-24

serial, parallel, super, quatam comp

Timing and Control: Control unit organization.

Hardwired Organization: Control logic is implemented

with gates, flip flops, decoders and other DC's.

Advantage: It can be optimized to produce fast mode of operation.

MICROPROGRAMMED ORGANIZATION:

- Control information is stored in a CONTROL MEMORY control memory is programmed to INITIATE the required SEQUENCE OF MICRO OPERATIONS.

→ Hardward control requires changes in wiring among the various components if the design has to be modified or changed.

→ In the microprogrammed control, any required changes or modifications can be done by UPDATING the MICRO PROGRAM in CONTROL MEMORY.

CONTROL UNIT: Block diagram has 3 parts.

a) TWO DECODERS b) SEQUENCE COUNTER
c) a number of CONTROL LOGIC GATES.

- An instruction read from memory is placed in the instruction REGISTER (IR) in Fig 5-6.

It is divided into 3 parts: the I bit, the opcode, and bits '0' through '11'.

opcode → [3 × 8 Decoder] →

The opcode in bits 12 through 14 are decoded with 3x8 decoder. The 8 outputs of the decoder are designated by the symbols 'D0' through 'D7'.

The subscribed decimal number is equivalent to the binary value of the corresponding operation code. Bit 15 of the instruction is transferred to a flip-flop designated by the symbol I. Bits 0 through '11'

are applied to the control logic gates

- The 4 bit sequence counter can count in binary from 0 through 15.
- The outputs of the counter are decoded into 16 timing signals to through T_{15}.
- The sequence counter can be incremented or cleared synchronously. Most of the time the counter is incremented to provide the sequence of timing signals out of 4x16 decoder.
- Once in a while, the counter is cleared to '0' causing the next active timing signal to be T_0.

EXAMPLE: Consider the case where SC is incremented to provide timing signals T_0, T_1, T_2, T_3, T_4 in sequence At the time T_4, SC is cleared to '0'. if decoder output D_3 is active. This is expressed symbolically by the statement

$$D_3 T_4 : SC \leftarrow 0$$

The timing diagram (fig. 5.7) shows the time relationship of the control signals.

- The SC responde to the positive transition of the clock
- Intially, the CLR input of SC is active
- The first positive transition of the clock clears sequence counter to '0', which intern activates the TIMING SIGNAL T_0 output of the decoder, SC

T_0 is active during one clock cycle.

The positive clock transition labelled T_0 in the diagram will trigger only those registers whose control inputs are connected to the timing signal T_0.

April 9

RISC	CISC
→ Reduced instruction set computing	→ Complex instruction set computing.
→ less number of instructions	→ largest set of instructions.
→ Few Addressing modes	→ Large no. of Addressing modes.
→ clock pulses are less for each instruction	→ clock pulses are more in number.
→ Impl of pipelining is easy	→ pipelining impl is difficult.

⇒ pipelining is the process of fetching memory when one is executing.

→ RISC pipeline

1. Latency: time taken to complete one instruction.
2. Throughput: No. of instructions completed / sec
3. Clock cycles:
4. CPI (clocks per instruction)
5. Speedup: Amount of time it takes in non-pipeline by pipeline

$= \frac{240}{160} = 1.5$

32-Register / 32 bits or 64 bits

→ RISC uses 5 stage Pipeline.

IF, ID, EX, MEM, WB

3 types of instru

ALU | LD & SD | Branch

Load & store | Jumps

Instru fetch | Inst decode | Inst execu | obtain address | write back

these program counter needs to be updated

and store memory.

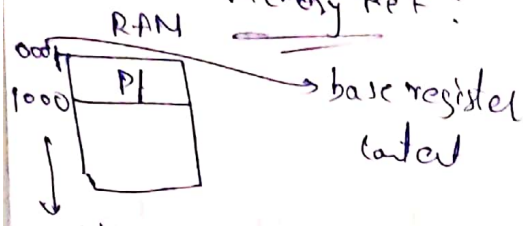
→ Automatic, static, external, register storage classes.

Instruction fetching: Move PC to address register and bring add register and store in IR and increment PC ⇝ PC ← PC + 4.

ID: 12, 13, 14 bits passed to decoder.

Ex: Regis Ref: ALU applies opes immediately and stored in another register.

Memory Ref: we need to calculate Base register + contents of offset.



Limit register ~ from base to limit how much we need to add is called offset.

→ Every program is protected by base, limit

→ ADD R3 10H, 09H (Register immediate)

ADD R3 R1, 100H (both registers)

IF ID EX MEM WB

IF ID EX MEM WB → In a non pipeline system it takes 30 cycles

IF ID EX MEM WB

1 2 3 4 5 6 7 → RISC-total-7

→ what happens if pipeline process struck up! jump instrns ??

draw backs of pipeline?

Hazards

→ Structural, Data, Control

Structural: occurs due to resource conflicts

Data: occurs due to change in order of instructions.

Control: occurs due to branch instructions. → jump {if else type

10th April

→ The first positive transition of the clock clears SC to 0' which inturn reactivates the timing signal T0 out of the decoder (4x16).

→ T0 is active during one clock cycle. The positive clock transition labelled in the diagram (5.7) will trigger only those register whose control inputs are connected to the timing signal, T0.

→ Generally SC is incremented with every +ve clock transition, unless its CLR input is active. This produces the sequence of timing signals T0, T1, T2, T3, T4 and so on (as shown in the diagram).

relationship b/w timing signals & its corresponding positive clock transition.
If SC is not cleared, the timing signal will continue with $T_5, T_6, \ldots$ up to T_{15} and back to T_0.
The last 3 waveforms in Fig. 5.7 show how SC is cleared when $D_3T_4 = 1$
The output D_3 from the OPERATION DECODER becomes active at the end of timing signal T_2
When the timing signal T_4 becomes active, the output of the AND Gate that implements the control function D_3T_4 becomes active. So the output signal is applied to the CLR input of the SC. On the next positive clock transition (the one marked T_4 in the diagram), the Sequence Counter is cleared to 0. So this causes the timing signal T_0 to become active, instead of T_5 that would have been active if SC were incremented instead of cleared.
- A memory read (or) write cycle will be initiated with the rising edge of a timing signal.
- A memory read (or) write cycle initiated by a timing signal will be completed by the time the next clock goes through its positive transition.
- The next clock transition will then be used to load the memory word into the register.

- The timing relationship is not valid in many computers, because the "memory cycle time" is usually longer than the processor clock cycle.
- In such case it is necessary to provide wait cycles in the processor until the memory word is available.
- To facilitate the presentation, we will assume that a WAIT PERIOD is not necessary in the basic computers.

→ To fully operand the operation of the computer it is crucial that one understands the timing relation ship between the clock transition & the timing signals.
For example the register transfer statement (control function)
$T_0: AR \leftarrow PC$ Specifies a transfer of content of PROGRAM COUNTER into address Register. if TIMING SIGNAL T_0 is ACTIVE. T_0 is active during the entire clock cycle interval.
During this time the content of PC is placed on to the "BUS" (with $S_2, S_1, S_0 = 010$) & the LD input of AR is enabled.
The Actual transfer does not occur until the end of clock cycle when the clock goes through a positive transition.

-) This same +ve clock transition increments the SC from 0000 to 0001. The next clock cycle has T_1 active & T_0 inactive

Instruction cycle:

- A program residing in the memory unit of a computer is executed in the computer by going through a sequence of instructions. The program cycle for each instruction

1 - Each instruction cycle in turn is subdivided into a sequence of subcycles (or) phases.

1 - In the basic computer each instruction cycle consists of the following phases/ subcycles

① Fetch & instruction

April 12 - Massbunk.

April 15

INSTRUCTION CYCLE: The micro operation for the FETCH & DECODE phases can be specified by the following REGISTER TRANSFER statements.

T_0 : AR ← PC
T_1 : IR ← M[AR], PC ← PC + 1
T_2 : $D_0, D_1, \ldots D_7$ ← DECODE IR(12-14), AR ← IR(0-11), I ← IR(15)

Fig 5.8 shows how the first two Register Transfer statements are implemented in the bus system to provide data path for the transfer of PC to AR we must apply timing signal T_0 to achieve the following connection.

Address bus is same as data bus.

1. Place the content of PC onto the bus by making the bus selection inputs $S_2 S_1 S_0 = 010$.
2. Transfer the content of the bus to AR by enabling the LD (load) input of AR.

→ The next clock transition initiates the transfer from PC to AR since $T_0 = 1$

DLD - continue

April-15

→ In order to implement the second statement

T1 : IR ← M[AR], PC ← PC+1 it is necessary to use timing signal T_1 to provide the following connections in the bus system.

i) Enable the Read input of memory

ii) Place the content of memory onto the bus by making $S_2 S_1 S_0 = 111$

iii) Transfer the content of the bus to IR by enabling the LD input of IR.

iv) Increment PC by enabling the INR input of PC.

The next clock transition initiates the read and increment operations since $T_1 = 1$

→ Fig 5.8 → its not entire bus & it doesn't contain all registers

→ It duplicates a PORTION of the bus system and shows how T_0 and T_1 are connected to the control inputs of the registers, the memory and the bus selection inputs.

→ Multiple input OR gates are included in the diagram because there are other control functions that will initiate similar operations.

April-16

Memory organization

→ Illegal jumps → control hazards.

Memory: storage area used to store programs and data to perform specific task. (Auxiliary, Main, Cache)

Auxiliary Memory: Not directly connected to CPU
Secondary storage devices
ex- tapes, discs.

Main Memory: RAM, ROM.

Cache Memory:

RAM: RANDOM ACCESS MEMORY

↳ facilitates direct access to a memory cell

SRAM	DRAM
→ stores one bit in flipflop.	→ one bit of information in a Transistor/capacitor
→ As long as power is supplied data is alive	→ very large in size
→ Generally used to create Cache-memory	→ Requires refreshing
→ Faster & expensive	→ used to create main memory
	→ slower & cheaper.

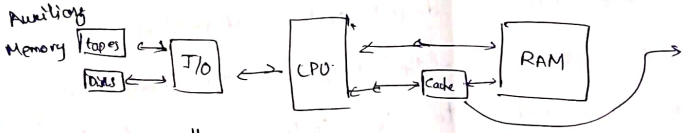
ROM: Read-only Memory

↳ program which are to be made permanent are stored in ROM

→ Bootstrap loader

- The moment we switch on power button to screen getting on is done by bootstrap loader.

Loader initiates the execution i.e loads the object code.



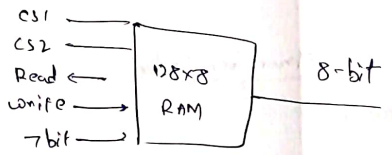
→ Cache performance is measured in terms of Hit ratio.

if hits a word in Cache then it's called hit or otherwise first it access in main memory of any data.

Cache memory

RAM is the most suitable memory to communicate with the CPU, if one can map control i/p's are provided.

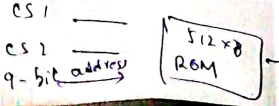
→ 128 words RAM & each word is of 1 byte



CS1	CS2	R	W	
1	0	0	1	write
1	0	1	0	Read
1	0	0	0	Inhibit

ROM- 512 word ROM

-one each word - 8-bit. → How to map main memory with Cache memory



- this mapping can be done in 3 ways (Associative, direct, Set-Associative)

April 17

T_0 : $AR \leftarrow PC$

T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

T_2 : $D_0, D_1, \dots D_7 \leftarrow$ DECODE $IR[12-14]$, $AR \leftarrow TR[0-11]$, $I \leftarrow IR(15)$

SEQUENTIAL COMPUTER
PARALLEL COMPUTER

Determine the Type of Instruction:

- During time T_3, the control unit determines the type of instruction that was just read from memory. The flow chart of Fig 5.9 presents an intial configuration for the INSTRUCTION CYCLE and it shows how the control determines the instruction type after the decoding. (There are 3 possible instruction types):

  - Memory Reference instructions
  - Register Reference instructions
  - I/O instructions

→ Decoder output D_7 is equal to '1' if the OPERATION CODE is equd to binary bit . But if $D_7 = 1$, the instruction must be a REGISTER REFERENCE OR INPUT-OUTPUT type

- If $D_7 = 0$, the OPERATION CODE must be ONE of other seven values 000 through 110, specifying a MEMORY REFERENCE INSTRUCTION.

- CONTROL then inspects the value of first bit of the instruction which is now available in flipflop

If $D_7 = 0$ and $I = 1$, we have a MEMORY REFERENCE instruction with an INDIRECT ADDRESS (we send the least significant) $AR \leftarrow M[AR]$

It is then necessary to read the effective address from memory. The MICRO-OPERATION for the INDIRECT ADDRESS condition can be symbolized by the REGISTER TRANSFER statement $AR \leftarrow M[AR]$.

Intially AR holds the address part of the instruction

This address is used during MEMORY READ. The word at the address given by AR is read from memory and placed on the common bus. The LD (load) input of AR is then enabled to receive the indirect address that resided in the 12 least significant bits of the memory word.

REGISTER REFERENCE INSTRUCTION

- Recognized by the control when $D_7 = 1$ and $I = 0$.

These instructions use bits '0' through '11' of the instruction code to specify one of 12 instructions. These 12 bits are available in IR [0-11]. They were also transferred to AR during time T_2.

April 22

Operation of Memory STACK (Storage Device):-

item in the STACK.

Contrary to stack of trays where the tray itself maybe taken out or inserted, the physical registers of a stack are always available for reading or writing. It is the content of the word that is inserted or deleted.

The two operations of a stack are the INSERTION and DELETION of items. The operation of insertion is called (PUSH) (or PUSHDOWN) because it can be thought of the result of pushing a new item on top

The operation of deletion is called POP (or POP up) because it can be thought of as the result of REMOVING one item. So that stack pops-up.

REMOVING one item so that stack pops-up
- CRUCIAL DIFFERENCE

However nothing is "pushed" or "poped" in a computer stack these operations are simulated by "INCREMENTING" or "DECREMENTING" the STACK POINTER REGISTER.

REGISTER STACK: A stack can be placed in a portion of large memory or it can be organised as a collection of a finite number of memory words or register.

Fig 8.3. ORGANIZATION of a 64-word Register stack.

SP (Stack Pointer Register): Contains the binary number whose value is equal to the address of the word that is currently on top of the stack.

Three items are placed in the stack: A, B, C in that order item C is on the top of the stack so that the content of SP is now 3. To remove the top item, the stack is popped by reading the memory word at address 3. Decrementing the Content of the stack pointer. Item B is now on top of stack since SP holds address 2.

→ To insert a new item, the stack is pushed by incrementing SP and WRITING a word in the next higher location in the stack.

Note: Item C has been read out but not physically removed This doesnot matter because when stack is pushed, a new item is written in its place.

MEMORY STACK

- A stack can exsist as a stand alone unit as in the case of Fig 8.3 or can be implemented in a RAM attached to CPU

- The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a STACK POINTER.
- Fig 8.4 shows a portion of Computer memory partitioned into 3 segments: Program, data, stack.

As shown in fig 8.4, the initial value of SP is 4001 and the stack grows with decreasing address.

Thus the first item stored in the stack is at address 4000, the second item is stored at address 3999, and the last address that can be used for the stack is 3000.

Register stack : FULL, EMPTY

stack limit checks: FULL : one bit register is set to 1
EMPTY : set to 1, when stack is empty. } check process stack

- we assume that the items in the stack communicate with a DATA REGISTER, (DR)

- A new item is inserted with the PUSH operation as follows:

SP ← SP-1
M[SP] ← DR

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

~~SP ← SP+1~~
DR ← M[DR]
SP ← SP+1

computer works on principle garbage in & garbage out.

- A new item is deleted with a pop-operation as follows

DR ← M[SP] = top item on the stack read from
SP ← SP+1 = stack pointer is incremented to point at the top. } stack to data register

- No hardware checks for stack overflow (full stack) or underflow (empty stack)

- Stack limits could be checked by 2 processor registers : one to hold upper limit & the other for lower limit.

Memory { - How much (size of memory)
- How fast (speed of memory write/Read)
- ... expensive:

Memory Hierarchy: CACHE MEMORY : very fast but costly

MAIN MEMORY: Medium

(secondary) AUXILARY MEMORY

CACHE { L1, L2, L3

INSTRUCTION PIPELINING:

- It is a technique for implements INSTRUCTION LEVEL PARALLELISM within a single processor. PIPELINING attempts to keep every part of the processor busy with some instruction by dividing incoming INSTRUCTIONS into a SERIES of SEQUENTIAL STEPS ("pipeline") performed by DIFFERENT PROCESSOR UNITS with different parts of instructions processed in parallel.

Example Processors

Atmel AVR, PIC Microcontroller.

Intel Pentium 4 (pipeline size: 7, 10 or even 20 stages)

Prescott & Ledar Mill Netburst cores from Intel.

April 26

Reduced Instruction Set Computer Architecture (RISC)

Microprocessor ERA:

- Computer scientists were trying to copy the same COMPLEX instructions used in Mainframe → 1950's computers. In fact, some wanted to expand those instructions.

- patterson & Hennesy, Idea: Not only should we NOT MAKE (instruction set) it more complicated, we should make it even simpler: called DANGEROUS IDEA.

- RISC architecture relied on a simpler collection of general functions the processor would perform shrinking the number of transistors needed to carry out a task.

- Today, 99% of more than 16 billion MICRO PROCESSORS produced each year are based on RISC architecture, powering smartphones, Tablets & IOT devices

everything in the world connected to internet.

Note:- It wasn't that the number of instructions was necessarily smaller, it was that they were less

→ those early computers had perhaps 25% more instructions "complex" under RISC design, but ran 5 times as fast.

(Example: BERKELEY RISC and STANFORD MIPS processor)

MIPS: Microprocessor without interlocked pipeline stages -- Hennessy) RISC architecture.

which speeded up processing by using instructions loaded from the memory into a register which could be accessed faster.

- DOMAIN SPECIFIC ARCHITECTURES for "ML".

SUPERSCALAR PROCESSOR (SSP)

SP is a CPU that implements a form of parallelism called INSTRUCTION LEVEL PARALLELISM within a SINGLE PROCESSOR.

- In contrast to a scalar processor that can execute at most one single instruction per clock cycle, a superscalar processor can execute MORE THAN one instruction during a clock cycle by SIMULTANEOUSLY DISPATCHING multiple instructions to DIFFERENT EXECUTION UNITS on the same processor.

- It therefore allows for MORE THROUGHPUT (the number of instructions that can be executed in a unit of time) than would otherwise be not possible at a give clock rate.

[they are on the same processor]

Each Execution unit is not a seperate processor (or a core if the processor is a multi core processor) but an EXECUTION RESOURCE within a single CPU such as ARITHMETIC & LOGIC UNIT

Single core super scalar Processor : SISD.

30/April 2019

Tutorial

Data Transfer model

I/O ↔ CPU ↔ Memory

1) programmed I/O
2. Interrupt - Intiate
3) Direct Memory Acces

Programmed I/O.

→ It is the result of execution of I/O instruction written in a program
→ CPU is kept in program loop and write an I/O device to transfer data.
→ Constant monitoring of I/O device is required by CPU

Interrupt - Intiate I/O:

→ processor with continuous task
→ whenever an I/O device is ready to transfer data, an interrupted is initiated to the CPU
→ Then the CPU have normal execution in attends to the then ISR (Interrupt Service routine)

→ hardware interrupt - Interrupt initiated by hardware device
→ S/w interrupt - ✓ software interrupt that occur during the program execution (RST) ✓ Reset

Maskable Interrupt / Non-Maskable Interrupt

↓ ↓

it is avoidable non-avoidable

→ 2 Model of DMA transfer:

1) cycle transfer 2) Burst transfer

✓ ✓

Buses will go back to the control of CPU after the transfer of every word

Bulk into exchange b/w I/O & memory in 1 bus grant itself

3/May 2019 1st class

CACHE MEMORY: ORGANIZATION of ASSOCIATIVE MAPPING:

property of Locality of Reference:

- Analysis of large number of typical programs has shown that the "REFERENCES to MEMORY" at any given interval of time tend to be confined within a few localised areas in memory.

This phenomenon is known as PROPERTY OF LOCALITY of Reference.

- The reason for this property may be understood by considering that a typical computer program flows in a straightline fashion with PROGRAM LOOPS and SUBROUTINE /FUNCTION CALLS ENCOUNTERED frequently.

- when a loop is executed, the CPU repeatedly refers to the set of instructions in memory that constitute

- Every time a given SUBROUTINE /FUNCTION is called, its set of instructions are fetched from memory

- Thus loops and subroutines tend to localise the

references to memory for fetching instructions.

- To lesser degree, memory references to data also tend to be localised. Table look-up procedures repeatedly refers to that portion in memory where the table is stored
- Iterative procedures refer to common memory location and array of numbers are confined within a local portion of memory.
- THE RESULT OF ALL THESE OBSERVATIONS is the locality of REFERENCE PROPERTY which states that over a short interval of time, the addresses generated by a typical program refer to a few localized areas of memory repeatedly while the remainder of memory is accessed relatively infrequently.

IDEA: If the active portions of the program and data are placed in a FAST, SMALL MEMORY, the AVERAGE MEMORY ACCESS TIME can be reduced, thus reducing the total execution time of the program

MIPS (millions of instructions per second) SUCH A FAST, small memory is referred to as a CACHE MEMORY

The CACHE MEMORY access time is less than the access time of MAIN MEMORY by a factor of 5-10. The cache is the fastest component in Memory Hierarchy and approaches the speed of CPU components. (eg: register)

access time will approach the access time of cache. Although the cache is only a small fraction of the size of main memory, a large fraction of memory requests will be found in the fast cache memory because of the locality of reference property of programs.

- The basic operation of cache is as follows. When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory. If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word. A BLOCK of words containing the one just accessed is then transferred from main memory to cache memory. The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed. In this manner, some data are transferred to the cache. So that future reference to memory find the required words in FAST CACHE MEMORY.

3/May/2019 2nd class.

HIT Ratio: The performance of cache memory is frequently measured in terms of a quantity called HIT RATIO. when the CPU refers to memory and finds the word in cache, it is said to produce a HIT.

- when the CPU refers to memory and finds the word in cache, it is said to produce a HIT.
- If the word is not found in cache, it is in main memory and it counts as a MISS.
- The ratio of the number of hits divided by the

total CPU references to memory (hits plus misses) is the HIT Ratio. (trade off between memory)

- The hit ratio is best measured experimentally by running representative programs in the computer and measuring the number of hits and misses during a given interval of time.

- Hit ratio of 0.9 and higher have been reported This high ratio verifies the validity of the LOCALITY OF REFERENCE PROPERTY.

The average memory access time of a computer System can be improved considerably by use of a cache. If the hit ratio is high enough so that most of the time the CPU accesses the cache instead of main memory, the average access time is closer to the access time of fast cache memory.

- The basic characteristic of CACHE MEMORY is its FAST ACCESS TIME. Therefore VERY LITTLE TIME or NO TIME must be wasted when searching for words in cache.

→ THE TRANSFORMATION of DATA FROM MAIN MEMORY TO CACHE MEMORY is referred to as a MAPPING PROCESS.

- 3 types of mapping procedures are of practical interest. when considering the organization of cache memory.
  1. ASSOCIATIVE MAPPING
  2. DIRECT MAPPING
  3. SET ASSOCIATIVE MAPPING.

- To help in discussion of these 3 mappings procedures we will use a specific example of MEMORY ORGANISATION as shown in Fig 12.10
  - The main memory can store 32K words of 12 bits each
  - The cache is capable of storing 512 of those words at any given time.
  - For every word stored in cache, there is a duplicate copy in main memory.
  - The CPU communicates with both memories. It first sends a 15-bit address to cache

If there is a hit, the CPU accepts the 12-bit data from cache. If there is a MISS, the CPU reads the word from main memory and the word is transfered — to cache

speed: cache L-1 ↑ cache L-2

2^{10} Main Memory: 32K × 12 → of 12 bits each.

$32 \times 2^{10} \rightarrow 2^{15}$ → total size.

- The CPU communicates with both memories it first sends a 15-bit address to cache. If there is a hit, the CPU accepts the 12-bit data from cache.
- If there is a miss, the CPU reads the word from main memory and the word is then transfered to cache.

ASSOCIATIVE MAPPING: The FASTEST and MOST FLEXIBLE cache organization uses an Associative memory.

- The AM stores both the address and content of the memory word.

THIS PERMITS ANY LOCATION IN CACHE TO STORE ANY WORD FROM MAIN MEMORY. The diagram shows three words presently stored in cache (12-11)

- The address value of 15 bits is shown as a FIVE

DIGIT OCTAL number and its corresponding 12-bit word is shown as FOUR DIGITAL octal number.

- A CPU address of 15 bits is placed in the "argument" register and the associative memory is searched for a matching address. If the address is found, the corresponding 12-bit data is read and sent to CPU.

— If no match occurs the main memory is accessed for the word.

- The address-data pair is then transferred to the ASSOCIATIVE CACHE MEMORY. If the cache is full, the address-data pair must be displaced to make room for a pair that is needed and not presently in the cache.

—The decision as to which pair is replaced is determined from the replacement algorithm that the designer chooses for cache

— A simple procedure is to replace cells of the cache in a ROUND ROBIN order whenever a new word is requested from the main memory. This constitutes a FIRST-IN-FIRST-OUT replacement policy.

→ How do we divide work to the processors?

RAM → serial address

DIRECT MAPPING: Normally AM's are expensive compared to RAM because of the added logic associated with each cell. The possibility of using RAM for the cache in Fig 12.12

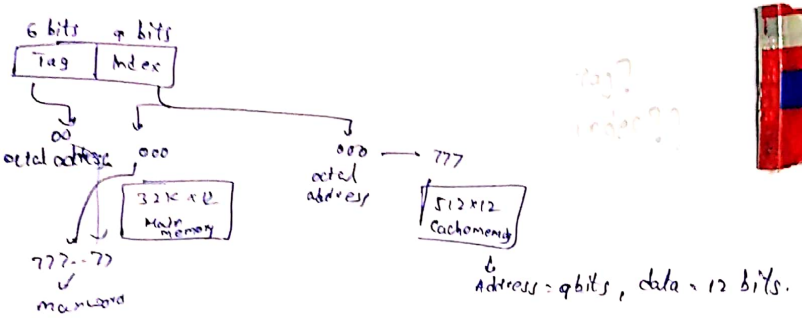
- The CPU address of 15 bits is divided into 2 fields the nine least significant bits constitute the INDEX FIELD and the remaining 6 bits form the TAG field.

- the figure shows that the MAIN MEMORY needs an address that includes both the TAG and the INDEX bits. The number of bits in the index field is equal to the number of address bits required to access cache memory

- In the general case, there are 2^k words in CACHE memory and 2^n words in main memory ($k < n$)
  - The n bit address is divided into 2 fields:
    K bits for the index field and n-k bits for the TAG field.

- The DIRECT MAPPING cache organization uses the n-bit address to access the main memory and k-bit index to access the cache

- The internal organization of the words in cache memory is as shown in fig-12.13(a)



- Each word in cache consists of the data word and its associated tag.
- When a new word is first brought into cache, the tag bits are stored alongside the data bits.
- When CPU generates a memory request, the index field is used for the address to access the cache.
- The TAG field of the CPU address is compared to with the tag in the word read from cache. If the two tags match, there is a hit and the desired data word is in cache

- If there is no match there is a MISS and the required word is read from main memory. It is then stored in cache together with new tag replacing the previous value.