INTROCTION ABOUT CPU, REGISTERS AND MEMORY, GENERAL REGISTER ORGANIZATION

A central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.

A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual characters). Some instructions specify registers as part of the instruction. For example, an instruction may specify that the contents of two defined registers be added together and then placed in a specified register.

Registers are the most important components of CPU. Each register performs a specific function. A brief description of most important CPU's registers and their functions are given below:

1. **Memory Address Register (MAR):** This register holds the address of memory where CPU wants to read or write data. When CPU wants to store some data in the memory or reads the data from the memory, it places the address of the required memory location in the MAR.

2. **Memory Buffer Register (MBR):** This register holds the contents of data or instruction read from, or written in memory. The contents of instruction placed in this register are transferred to the Instruction Register, while the contents of data are transferred to the accumulator or I/O register. In other words you can say that this register is used to store data/instruction coming from the memory or going to the memory.

3. **Program Counter (PC):** Program Counter register is also known as Instruction Pointer Register. This register is used to store the address of the next instruction to be fetched for execution. When the instruction is fetched, the value of IP is incremented. Thus this register always points or holds the address of next instruction to be fetched.

4. **Instruction Register (IR):** Once an instruction is fetched from main memory, it is stored in the Instruction Register. The control unit takes instruction from this register, decodes and executes it by sending signals to the appropriate component of computer to carry out the task.

5. **Accumulator Register:** The accumulator register is located inside the ALU; it is used during arithmetic & logical operations of ALU. The control unit stores data values fetched from main memory in the accumulator for arithmetic or logical operation. This register holds the initial data to be operated upon, the intermediate results, and the final result of operation. The final result is transferred to main memory through MBR.

6. **Stack Control Register:** A stack represents a set of memory blocks; the data is stored in and retrieved from these blocks in an order, i.e. First In and Last Out (FILO). The Stack Control Register is used to manage the stacks in memory. The size of this register is 2 or 4 bytes.

7. **Flag Register:** The Flag register is used to indicate occurrence of a certain condition during an operation of the CPU. It is a special purpose register with size one byte or two bytes. Each bit of the flag register constitutes a flag (or alarm), such that the bit value indicates if a specified condition was encountered while executing an instruction.

For example, if zero value is put into an arithmetic register (accumulator) as a result of an arithmetic operation or a comparison, then the zero flag will be raised by the CPU. Thus, the subsequent instruction can check this flag and when a zero flag is "ON" it can take an appropriate route in the algorithm.

Memory

This unit can store instructions, data, and intermediate results. This unit supplies information to other units of the computer when needed. It is also known as internal storage unit or the main memory or the primary storage or Random Access Memory (RAM).

Its size affects speed, power, and capability. Primary memory and secondary memory are two types of memories in the computer. Functions of the memory unit are −

1. It stores all the data and the instructions required for processing.
2. It stores intermediate results of processing.
3. It stores the final results of processing before these results are released to an output device.
4. All inputs and outputs are transmitted through the main memory.

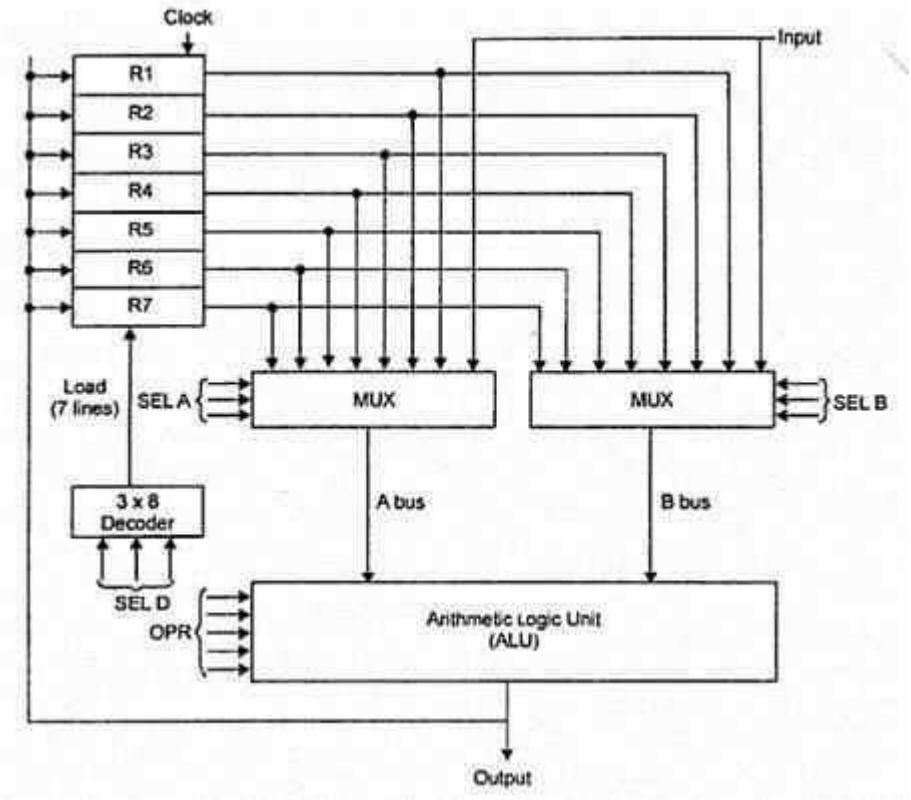
General Register organization

Generally CPU has seven general registers. Register organization show how registers are selected and how data flow between register and ALU. A decoder is used to select a

particular register. The output of each register is connected to two multiplexers to form the two buses A and B. The selection lines in each multiplexer select the input data for the particular bus.

The A and B buses form the two inputs of an ALU. The operation select lines decide the micro operation to be performed by ALU. The result of the micro operation is available at the output bus. The output bus connected to the inputs of all registers, thus by selecting a destination register it is possible to store the result in it.

A bus organization for seven CPU registers



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

Operation with symbol

Operation selection code	Operation	symbol
0000	Transfer A	TSFA
0001	Increment A	INC A
0010	A+B	ADD
0011	A-B	SUB
0100	Decrement A	DEC
0101	A AND B	AND
0110	A OR B	OR
0111	A XOR B	XOR
1000	Complement A	COMA
1001	Shift right A	SHR
1010	Shift left A	SHL

What is CONTROL WORD?

- The combined value of a binary selection inputs specifies the control word.
- It consists of four fields SELA, SELB, and SELD or SELREG contains three bit each and SELOPR field contains four bits thus the total bits in the control word are 13-bits.

SEL A	SELB	SELREG OR SELD	SELOPR

FORMATE OF CONTROL WORD

1. The three bit of SELA select a source registers of the input of the ALU.
2. The three bits of SELB select a source registers of the b input of the ALU.
3. The three bits of SELED or SELREG select a destination register using the decoder.
4. The four bits of SELOPR select the operation to be performed by ALU.

CONTROL WORD FOR OPERATION R2 = R1+R3

SEL A	SEL B	SEL D OR SELREG	SELOPR
001	011	010	0010

Note: Control words for all micro operation are stored in the control memory

Example:

MICROOPERATION	SEL A	SEL B	SEL D OR SELREG	SELOPR	CONTROL WORD			
R2 = R1+R3	R1	R3	R2	ADD	001	011	010	0010

DAY 3-10 STACK ORGANIZATION, INSTRUCTION FORMAT &ADDRESSING MODE

STACK ORGANIZATION

Stack is a storage structure that stores information in such a way that the last item stored is the first item retrieved. It is based on the principle of LIFO (Last-in-first-out). The stack in digital computers is a group of memory locations with a register that holds the address of top of element. This register that holds the address of top of element of the stack is called ***Stack Pointer***.

Stack Operations

The two operations of a stack are:

1. **Push:** Inserts an item on top of stack.
2. **Pop:** Deletes an item from top of stack.

Implementation of Stack

In digital computers, stack can be implemented in two ways:

1. Register Stack
2. Memory Stack

Register Stack

A stack can be organized as a collection of finite number of registers that are used to store temporary information during the execution of a program. The stack pointer (SP) is a register that holds the address of top of element of the stack.

Memory Stack

A stack can be implemented in a random access memory (RAM) attached to a 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. The starting memory location of the stack is specified by the processor register as *stack pointer*.

INSTRUCTION FORMATS

The physical and logical structure of computers is normally described in reference manuals provided with the system. Such manuals explain the internal construction of the CPU, including the processor registers available and their logical capabilities. They list all

hardware-implemented instructions, specify their binary code format, and provide a precise definition of each instruction. A computer will usually have a variety of instruction code formats. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction.

The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words or in a control register. The bits of the instruction are divided into groups called fields. The most common fields found in instruction formats are:

1. An operation code field that specifies the operation to be performed.
2. An address field that designates a memory address or a processor registers.
3. A mode field that specifies the way the operand or the effective address is determined.

Computers may have instructions of several different lengths containing varying number of addresses. The number of address fields in the instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of three types of CPU organizations:

1 Single accumulator organization.
2 General register organization.
3 Stack organization.

THREE-ADDRESS INSTRUCTIONS

Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand. The program in assembly language that evaluates X = (A + B) * (C + D) is shown below, together with comments that explain the register transfer operation of each instruction.

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

It is assumed that the computer has two processor registers, R1 and R2. The symbol M [A] denotes the operand at memory address symbolized by A.

The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions. The disadvantage is that the binary-coded instructions

require too many bits to specify three addresses. An example of a commercial computer that uses three-address instructions is the Cyber 170. The instruction formats in the Cyber computer are restricted to either three register address fields or two register address fields and one memory address field.

TWO-ADDRESS INSTRUCTIONS

Two address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word. The program to evaluate X = (A + B) * (C + D) is as follows:

MOV R1, A	R1 ← M [A]
ADD R1, B	R1 ← R1 + M [B]
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

The MOV instruction moves or transfers the operands to and from memory and processor registers. The first symbol listed in an instruction is assumed to be both a source and the destination where the result of the operation is transferred.

ONE-ADDRESS INSTRUCTIONS

One-address instructions use an implied accumulator (AC) register for all data manipulation. For multiplication and division there is a need for a second register. However, here we will neglect the second and assume that the AC contains the result of tall operations. The program to evaluate X = (A + B) * (C + D) is

LOAD A	AC ← M [A]
ADD B	AC ← A [C] + M [B]
STORE T	M [T] ← AC
LOAD C	AC ← M [C]
ADD D	AC ← AC + M [D]
MUL T	AC ← AC * M [T]
STORE X	M [X] ← AC

All operations are done between the AC register and a memory operand. T is the address of a temporary memory location required for storing the intermediate result.

ZERO-ADDRESS INSTRUCTIONS

A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack. The following program shows how X = (A + B) * (C + D) will be written for a stack organized computer. (TOS stands for top of stack)

PUSH A	TOS ← A
PUSH B	TOS ← B
ADD	TOS ← (A + B)
PUSH C	TOS ← C
PUSH D	TOS ← D
ADD	TOS ← (C + D)
MUL	TOS ← (C + D) * (A + B)
POP X	M [X] ← TOS

To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into reverse Polish notation. The name "zero-address" is given to this type of computer because of the absence of an address field in the computational instructions.

ADDRESSING MODES

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in computer registers or memory words. The way the operands are chosen during program execution in dependent on the addressing mode of the instruction. The addressing mode of the instruction. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Although most addressing modes modify the address field of the instruction, there are two modes that need no address field at all. These are the implied and immediate modes.

1 Implied Mode: In this mode the operands are specified implicitly in the definition of the instruction. For example, the instruction "complement accumulator" is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction. In fact, all register reference instructions that use an accumulator are implied-mode instructions. Instruction format with mode field Zero-address instructions in a stack-

organized computer are implied-mode instructions since the operands are implied to be on top of the stack.

2 Immediate Mode: In this mode the operand is specified in the instruction itself. In other words, an immediate mode instruction has an operand field rather than an address field. The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction. Immediate-mode instructions are useful for initializing registers to a constant value. It was mentioned that the address field of an instruction may specify either a memory word or a processor register. When the address field specifies a processor register, the instruction is said to be in the register mode.

3 Register Mode: In this mode the operands are in registers that reside within the CPU. The particular register is selected from a register field in the instruction. A k-bit field can specify any one of 2k registers.

4 Register Indirect Mode: In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. In other words, the selected register contains the address of the operand rather than the Op code Mode Address operand itself. Before using a register indirect mode instruction, the programmer must ensure that the memory address for the operand is placed in the processor register with a previous instruction. A reference to the register is then equivalent to specifying a memory address. The advantage of a register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.

5 Auto increment or Auto decrement Mode: This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory. When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table. This can be achieved by using the increment or decrement instruction. However, because it is such a common requirement, some computers incorporate a special mode that automatically increments or decrements the content of the register after data access. The address field of an instruction is used by the control unit in the CPU to obtain the operand from memory. Sometimes the value given in the address field is the address of the operand, but sometimes it

is just an address from which the address of the operand is calculated. To differentiate among the various addressing modes it is necessary to distinguish between the address part of the instruction and the effective address used by the control when executing the instruction. The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode. The effective address is the address of the operand in a computational-type instruction. It is the address where control branches in response to a branch-type instruction.

6 Direct Address Mode: In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction. In a branch-type instruction the address field specifies the actual branch address.

7 Indirect Address Mode: In this mode the address field of the instruction gives the address where the effective address is stored in memory. Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

8 Relative Address Mode: In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address. The address part of the instruction is usually a signed number (in 2's complement representation) which can be either positive or negative. When this number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction. To clarify with an example, assume that the program counter contains the number 825 and the address part of the instruction contains the number 24. The instruction at location 825 is read from memory during the fetch phase and the program counter is then incremented by one to 826 + 24 = 850. This is 24 memory locations forward from the address of the next instruction. Relative addressing is often used with branch-type instructions when the branch address is in the area surrounding the instruction word itself. It results in a shorter address field in the instruction format since the relative address can be specified with a smaller number of bits compared to the number of bits required to designate the entire memory address.

9 Indexed Addressing Mode: In this mode the content of an index register is added to the address part of the instruction to obtain the effective address. The index register is a special CPU register that contains an index value. The address field of the instruction defines the beginning address of a data array in memory. Each operand in the array is stored in memory

relative to the beginning address. The distance between the beginning address and the address of the operand is the index value stores in the index register. Any operand in the array can be accessed with the same instruction provided that the index register contains the correct index value. The index register can be incremented to facilitate access to consecutive operands. Note that if an index-type instruction does not include an address field in its format, the instruction converts to the register indirect mode of operation. Some computers dedicate one CPU register to function solely as an index register. This register is involved implicitly when the index-mode instruction is used. In computers with many processor registers, any one of the CPU registers can contain the index number. In such a case the register must be specified explicitly in a register field within the instruction format.

10 Base Register Addressing Mode: In this mode the content of a base register is added to the address part of the instruction to obtain the effective address. This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register. The difference between the two modes is in the way they are used rather than in the way that they are computed. An index register is assumed to hold an index number that is relative to the address part of the instruction. A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address. The base register addressing mode is used in computers to facilitate the relocation of programs in memory. When programs and data are moved from one segment of memory to another, as required in multiprogramming systems, the address values of the base register requires updating to reflect the beginning of a new memory segment.

CONCEPT OF CPU DESIGN, RISC & CISC

Day 12 to 18

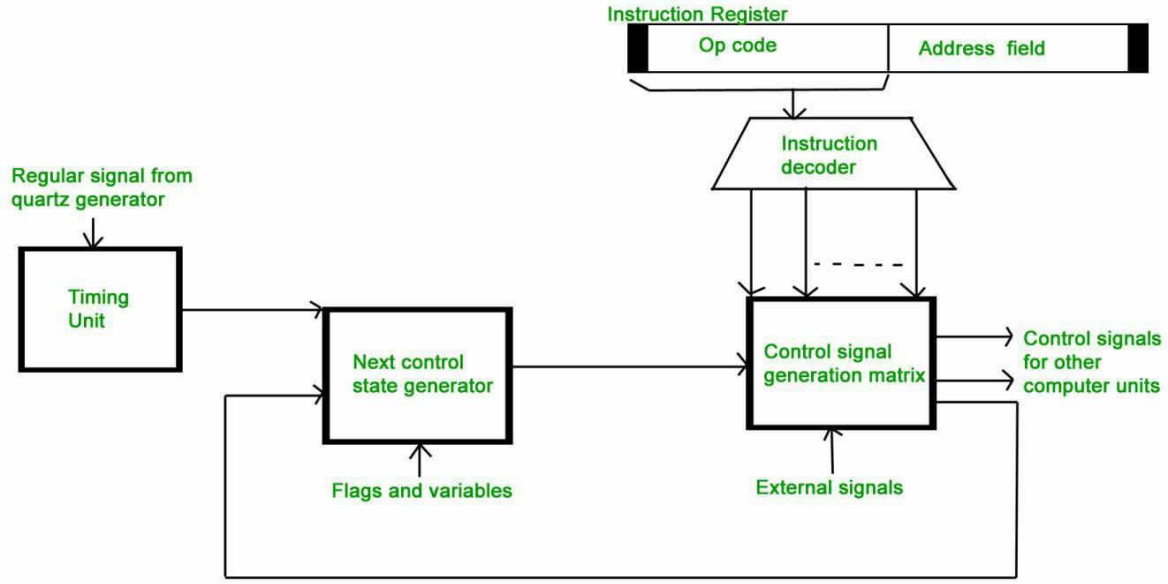
Hardwired v/s Micro-programmed Control Unit

To execute an instruction, the control unit of the CPU must generate the required control signal in the proper sequence. There are two approaches used for generating the control signals in proper sequence as Hardwired Control unit and Micro-programmed control unit.

Hardwired Control Unit –

The control hardware can be viewed as a state machine that changes from one state to another in every clock cycle, depending on the contents of the instruction register, the condition codes and the external inputs. The outputs of the state machine are the control signals. The sequence of the operation carried out by this machine is determined by the wiring of the logic elements and hence named as "hardwired".

1. Fixed logic circuits that correspond directly to the Boolean expressions are used to generate the control signals.
2. Hardwired control is faster than micro-programmed control.
3. A controller that uses this approach can operate at high speed.

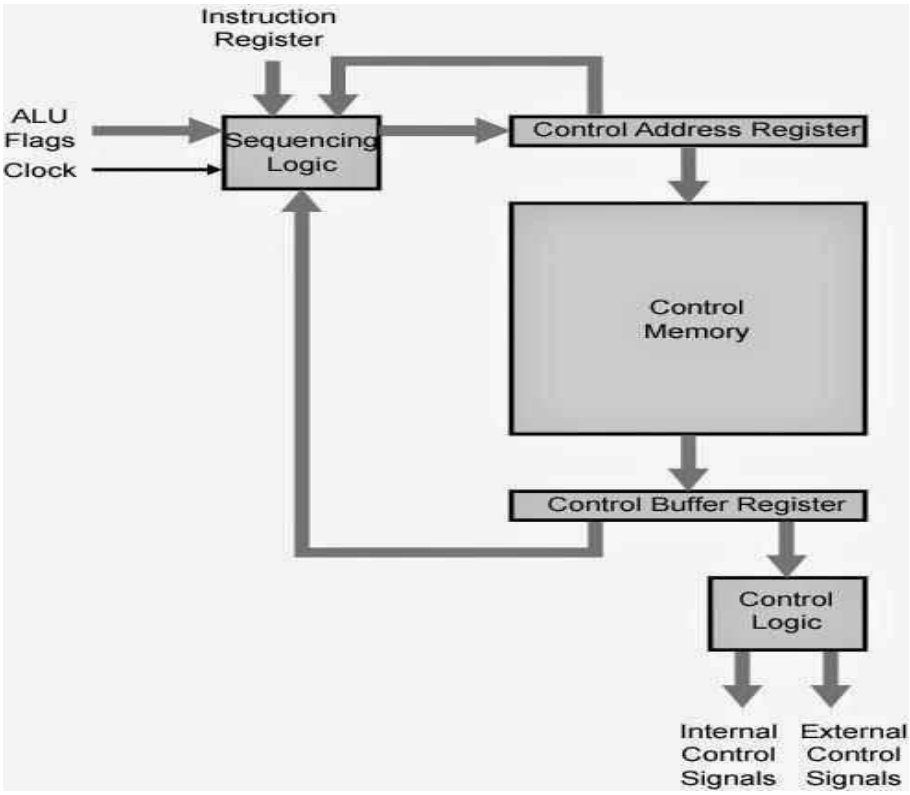


Characteristics:

1. It uses flags, decoder, logic gates and other digital circuits.
2. As name implies it is a hardware control unit.
3. On the basis of input Signal output is generated.
4. Difficult to design, test and implement.
6. Inflexible to modify.
7. Faster mode of operation.
8. Expensive and high error.
9. Used in RISC processor.

Micro-programmed Control Unit –

1. The control signals associated with operations are stored in special memory units inaccessible by the programmer as Control Words.
2. Control signals are generated by a program are similar to machine language programs.
3. Micro-programmed control unit is slower in speed because of the time it takes to fetch microinstructions from the control memory.



Characteristics

1. It uses sequence of micro-instruction in micro programming language.

2. It is mid-way between Hardware and Software.

3. It generates a set of control signal on the basis of control line.

4. Easy to design, test and implement.

5. Flexible to modify.

6. Slower mode of operation.

7. Cheaper and less error.

8. Used in CISC processor.

Reduced Instruction Set Computer

A reduced instruction set computer (RISC) is a computer that uses a central processing unit (CPU) that implements the processor design principle of simplified instructions. To date, RISC is the most efficient CPU architecture technology.

This architecture is an evolution and alternative to complex instruction set computing (CISC). With RISC, the basic concept is to have simple instructions that do less but execute very quickly to provide better performance.

The most basic RISC feature is a processor with a small core logic that allows engineers to increase the register set and increase internal parallelism by using the following:

1. Thread level parallelism: Increases the number of parallel threads executed by the CPU
2. Instruction level parallelism: Increases the speed of the CPU's executing instructions

The words "reduced instruction set" are often misinterpreted to refer to a reduced number of instructions. However, this is not the case, as several RISC processors, like the PowerPC, have numerous instructions. At the opposite end of the spectrum, the DEC PDP-8, a CISC CPU, has only eight basic instructions. Reduced instruction actually means that the amount of work done by each instruction is reduced in terms of number of cycles - at most only a single data memory cycle - compared to CISC CPUs, in which dozens of cycles are required prior to completing the entire instruction. This results in faster processing.

CISC

The main intend of the CISC processor architecture is to complete task by using less number of assembly lines. For this purpose, the processor is built to execute a series of operations. Complex instruction is also termed as MULT, which operates memory banks of a computer directly without making the compiler to perform storing and loading functions.

Features of CISC Architecture

1. To simplify the computer architecture, CISC supports microprogramming.
2. CISC have more number of predefined instructions which makes high level languages easy to design and implement.
3. CISC consists of less number of registers and more number of addressing modes, generally 5 to 20.
4. CISC processor takes varying cycle time for execution of instructions – multi-clock cycles.
5. Because of the complex instruction set of the CISC, the pipelining technique is very difficult.

6. CISC consists of more number of instructions, generally from 100 to 250.
7. Special instructions are used very rarely.
8. Operands in memory are manipulated by instructions.

Advantages of CISC architecture

1. Each machine language instruction is grouped into a microcode instruction and executed accordingly, and then are stored inbuilt in the memory of the main processor, termed as microcode implementation.
2. As the microcode memory is faster than the main memory, the microcode instruction set can be implemented without considerable speed reduction over hard wired implementation.
3. Entire new instruction set can be handled by modifying the micro program design.
4. CISC, the number of instructions required to implement a program can be reduced by building rich instruction sets and can also be made to use slow main memory more efficiently.
5. Because of the superset of instructions that consists of all earlier instructions, this makes micro coding easy.

Drawbacks of CISC

1. The amount of clock time taken by different instructions will be different – due to this – the performance of the machine slows down.
2. The instruction set complexity and the chip hardware increases as every new version of the processor consists of a subset of earlier generations.
3. Only 20% of the existing instructions are used in a typical programming event, even though there are many specialized instructions in existence which are not even used frequently.
4. The conditional codes are set by the CISC instructions as a side effect of each instruction which takes time for this setting – and, as the subsequent instruction changes the condition code bits – so, the compiler has to examine the condition code bits before this happens.

Difference between CISC and RISC

Architectural Characteristics	Complex Instruction Set Computer(CISC)	Reduced Instruction Set Computer(RISC)
Instruction size and format	Large set of instructions with variable formats (16-64 bits per instruction).	Small set of instructions with fixed format (32 bit).
Data transfer	Memory to memory.	Register to register.
CPU control	Most micro coded using control memory (ROM) but modern CISC use hardwired control.	Mostly hardwired without control memory.
Instruction type	Not register based instructions.	Register based instructions.
Memory access	More memory access.	Less memory access.
Clocks	Includes multi-clocks.	Includes single clock.
Instruction nature	Instructions are complex.	Instructions are simple.

DAY 21 TO 34

CONCEPT OF MEMORY ORGANIZATION

The memory unit is an essential component in any digital computer since it is needed for storing programs and Data. A very small computer with a limited application may be able to fulfill its intended task without the need of additional storage capacity. Most general-purpose computers would run more efficiently if they were equipped with additional storage beyond the capacity of the main memory. There is just not enough space in one memory unit to accommodate all the programs used in a typical computer. Moreover, most computer users accumulate and continue to accumulate large amounts of data-processing software. Not all accumulated information is needed by the processor at the same time. Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by the CPU. The memory unit that communicates directly with the CPU is called the main memory. Devices that provide backup storage are called auxiliary memory. The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large data files, and

other backup information. Only programs and data currently needed by the processor reside in main memory. All other information is stored in auxiliary Memory and transferred to main memory when needed.

Memory Hierarchy

Memory is categorized into volatile and nonvolatile memories, with the former requiring constant power ON of the system to maintain data storage. Furthermore, a typical computer system provides a hierarchy of different times of memories for data storage.

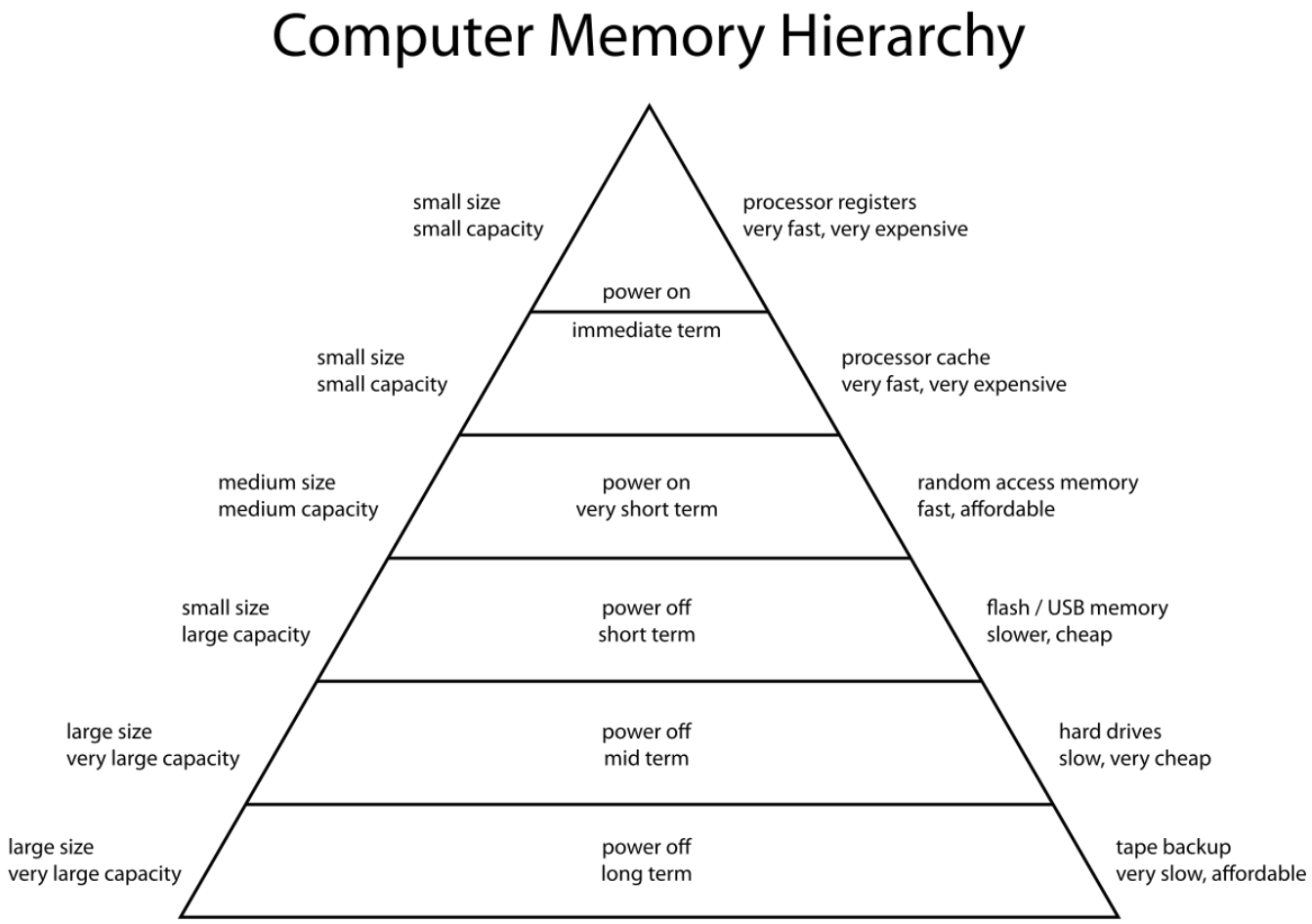
Different levels of the memory hierarchy

1. Cache (MB): Cache is the fastest accessible memory of a computer system. Its access speed is in the order of a few nanoseconds. It is volatile and expensive, so the typical cache size is in the order of megabytes.

2. Main memory (GB): Main memory is arguably the most used memory. When discussing computer algorithms such as quick sort, balanced binary sorted trees, or fast Fourier transform, one typically assumes that the algorithm operates on data stored in the main memory. The main memory is reasonably fast, with access speed around 100 nanoseconds. It also offers larger capacity at a lower cost. Typical main memory is in the order of 10 GB. However, the main memory is volatile.

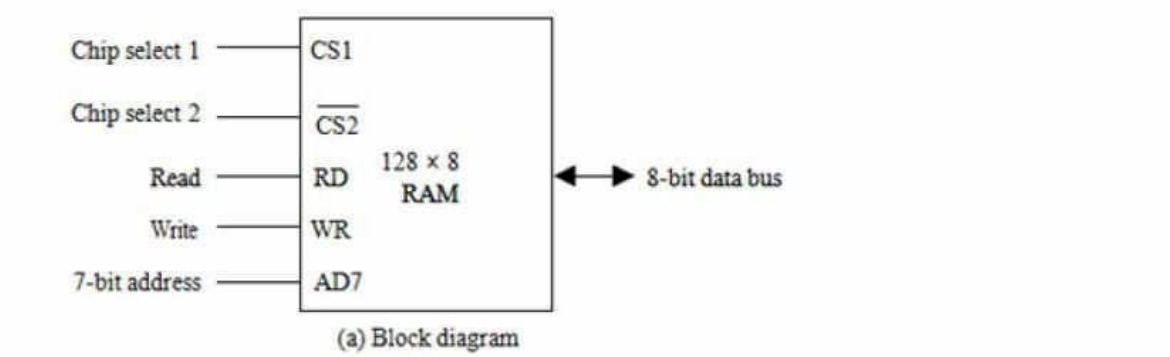
3. Secondary storage (TB): Secondary storage refers to nonvolatile data storage units that are external to the computer system. Hard drives and solid state drives are examples of secondary storage. They offer very large storage capacity in the order of terabytes at very low cost. Therefore, database servers typically have an array of secondary storage devices with data stored distributed and redundantly across these devices. Despite the continuous improvements in access speed of hard drives, secondary storage devices are several magnitudes slower than main memory. Modern hard drives have access speed in the order of a few milliseconds.

4. Tertiary storage (PB): Tertiary storage refers storage designed for the purpose data backup. Examples of tertiary storage devices are tape drives are robotic driven disk arrays. They are capable of peta byte range storage, but have very slow access speed with data access latency in seconds or minutes.



RAM AND ROM CHIPS

A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip only when needed. Another common feature is a bidirectional data bus that allows the transfer of data either from memory to CPU during a read operation or from CPU to memory during a write operation. A bidirectional bus can be constructed with three-state buffers. A three-state buffer output can be placed in one of three possible states: a signal equivalent to logic 1, a signal equivalent to logic 0, or a high-impedance state. The logic 1 and 0 are normal digital signals. The high-impedance state behaves like an open circuit, which means that the output does not carry a signal and has no logic significance. The block diagram of a RAM chip is shown in Fig. The capacity of the memory is 128 words of eight bits (one byte) per word.



(a) Block diagram

CSI	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	High-impedance
0	1	x	x	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	Output data from RAM
1	1	x	x	Inhibit	High-impedance

(b) Function table

This requires a 7-bit Address and an 8-bit bidirectional data bus. The read and write inputs specify the memory operation and the two chips select (CS) control inputs are for enabling the chip only when it is selected by the microprocessor. The availability of more than one control input to select the chip facilitates the decoding of the address lines when multiple chips are used in the microcomputer. The read and write inputs are sometimes combined into one line labeled R/W. When the chip is selected, the two binary states in this line specify the two operations or read or write.

The function table listed in Fig. (b) Specifies the operation of the RAM chip. The unit is in operation only when CSI = 1 and CS2 = 0. The bar on top of the second select variable indicates that this input in enabled when it is equal to 0. If the chip select inputs are not enabled, or if they are enabled but the read but the read or write inputs are not enabled, the memory is inhibited and its data bus is in a high-impedance state. When CS1 = 1 and CS2 = 0, the memory can be placed in a write or read mode. When the WR input is enabled, the memory stores a byte from the data bus into a location specified by the address input lines. When the RD input is enabled, the content of the selected byte is placed into the data bus. The RD and WR signals control the memory operation as well as the bus buffers associated with the bidirectional data bus.

A ROM chip is organized externally in a similar manner. However, since a ROM can only read, the data bus can only be in an output mode. The block diagram of a ROM chip is shown in below Fig. For the same-size chip, it is possible to have more bits of ROM occupy less space than in RAM. For this reason, the diagram specifies a 512-byte ROM, while the RAM has only 128 bytes. The nine address lines in the ROM chip specify any one of the 512 bytes stored in it. The two chip select inputs must be CS1 = 1 and CS2 = 0 for the unit to operate. Otherwise, the data bus is in a high-impedance state. There is no need for a read or write control because the unit can only read. Thus when the chip is enabled by the two select inputs, the byte selected by the address lines appears on the data bus.

MEMORY ADDRESS MAP

The designer of a computer system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM. The interconnection between memory and processor is then established form knowledge of the size of memory needed and the type of RAM and ROM chips available. The addressing of memory can be established by means of a table that specifies the memory address assigned to each chip. The table, called a memory address map, is a pictorial representation of assigned address space for each chip in the system.

To demonstrate with a particular example, assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM. The RAM and ROM chips

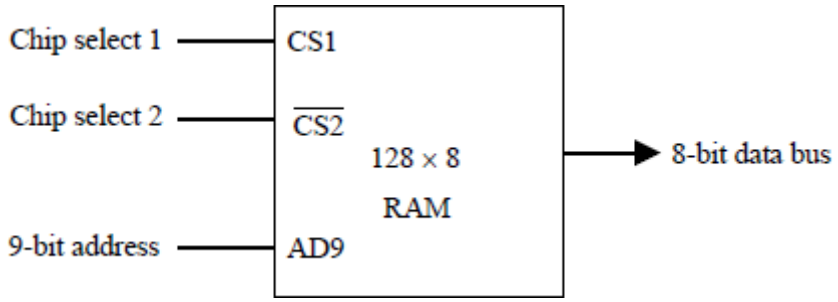


Figure-Typical ROM chip.

To be used are specified in Fig Typical RAM chip and Typical ROM chip. The memory address map for this configuration is shown in Table Below. The component column specifies whether a RAM or a ROM chip is used. The hexadecimal address column assigns a range of hexadecimal equivalent addresses for each chip. The address bus lines are listed in the third column. Although there are 16 lines in the address bus, the table shows only 10 lines because the other 6 are not used in this example and are assumed to be zero. The small x's

under the address bus lines designate those lines that must be connected to the address inputs in each chip. The RAM chips have 128 bytes and need seven address lines. The ROM chip has 512 bytes and needs 9 address lines. The x's are always assigned to the low-order bus lines: lines 1 through 7 for the RAM and lines 1 through 9 for the ROM. It is now necessary to distinguish between four RAM chips by assigning to each a different address. For this particular example we choose bus lines 8 and 9 to represent four distinct binary combinations. Note that any other pair of unused bus lines can be chosen for this purpose. The table clearly shows that the nine low-order bus lines constitute a memory space from RAM equal to 512 bytes. The distinction between a RAM and ROM address is done with another bus line. Here we choose line 10 for this purpose. When line 10 is 0, the CPU selects a RAM, and when this line is equal to 1, it selects the ROM. The equivalent hexadecimal address for each chip is obtained forms the information under the address bus assignment.

TABLE-Memory Address Map for Micro pro computer

Component	Hexadecimal address	Address bus		
		10 9	8 7 6 5	4 3 2 1
RAM 1	0000—007F	0 0	0 x x x	x x x x
RAM 2	0080—00FF	0 0	1 x x x	x x x x
RAM 3	0100—017F	0 1	0 x x x	x x x x
RAM 4	0180—01FF	0 1	1 x x x	x x x x
ROM	0200—03FF	1 x	x x x x	x x x x

The address bus lines are subdivided into groups of four bits each so That each group can be represented with a hexadecimal digit. The first hexadecimal digit represents lines 13 to 16 and is always 0. The next hexadecimal digit represents lines 9 to 12, but lines 11 and 12 are always 0. The range of hexadecimal addresses for each component is determined from the x's associated with it. This x's represent a binary number that can range from an all-0's to an all-1's value.

MEMORY CONNECTION TO CPU

RAM and ROM chips are connected to a CPU through the data and address buses. The low-order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through its chip select inputs. The connection of memory chips to the CPU is shown in below Fig. This configuration gives a memory capacity of 512 bytes of RAM and 512 bytes of ROM. It implements the memory map of Table above. Each RAM receives the seven low-order bits of the address bus to select one of 128 possible bytes. The particular RAM chip selected is determined from lines 8 and 9 in the address bus. This is done through a 2 × 4 decoder whose outputs go to the SCI input in each RAM chip. Thus, when address lines 8 and 9 are equal to 00, the first RAM chip is selected. When 01, the second RAM chip is selected, and so on. The RD and WR outputs from the microprocessor are applied to the inputs of each RAM chip. The selection between RAM and ROM is achieved through bus line 10. The RAMs are selected when the bit in this line is 0, and the ROM when the bit is 1. The other chip select input in the ROM is connected to the RD control line for the ROM chip to be enabled only during a read operation. Address bus lines 1 to 9 are applied to the input address of ROM without going through the decoder. This assigns addresses 0 to 511 to RAM and 512 to 1023 to ROM. The data bus of the ROM has only an output capability, whereas the data bus connected to the RAMs can transfer information in both directions.

The example just shown gives an indication of the interconnection complexity that can exist between memory chips and the CPU. The more chips that are connected, the more external decoders are required for selection among the chips. The designer must establish a memory map that assigns addresses to the various chips from which the required connections are determined.

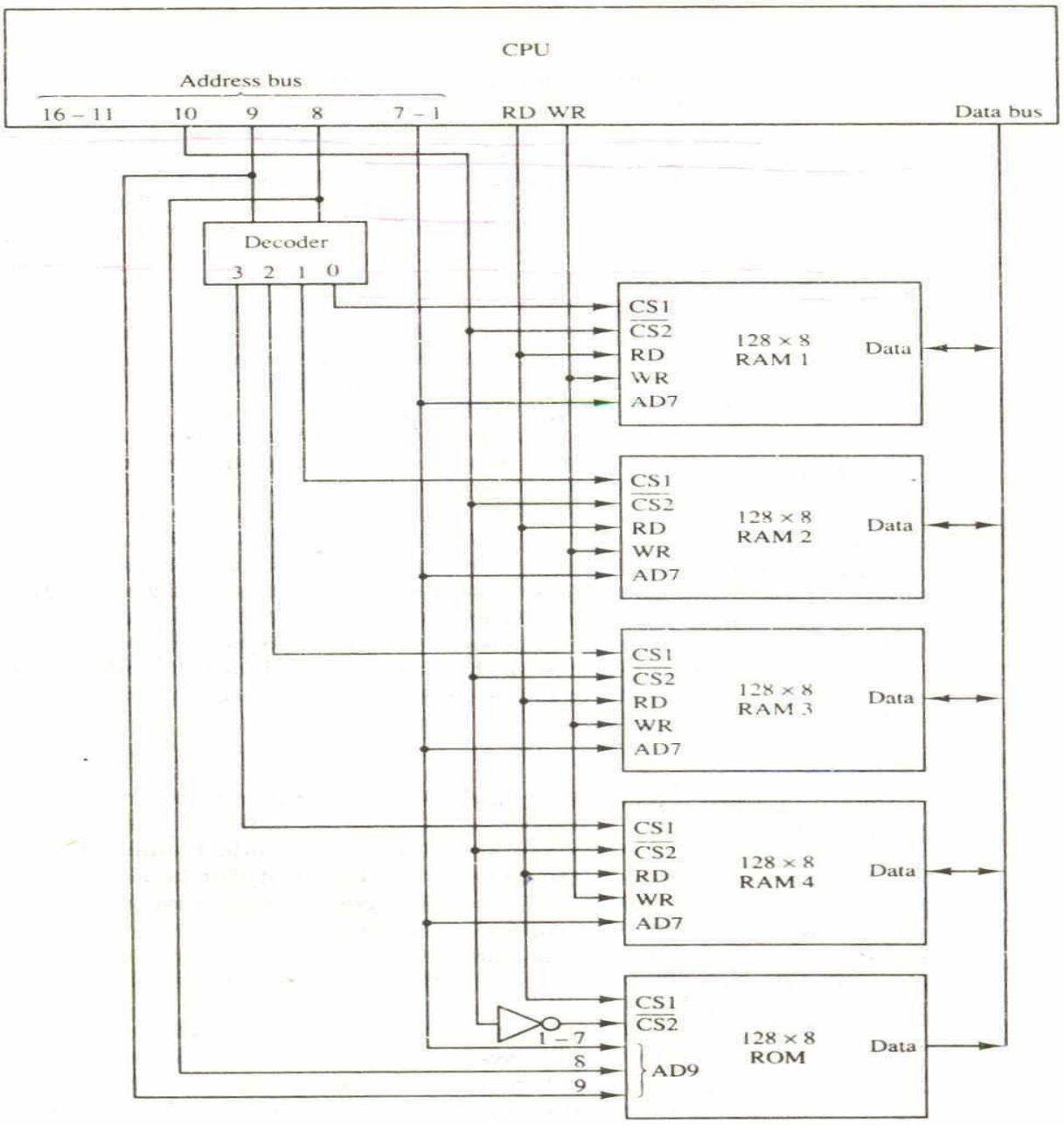


Figure -Memory connection to the CPU.

Auxiliary memory

(also referred to as secondary storage) is the non-volatile memory lowest-cost, highest-capacity, and slowest-access storage in a computer system. It is where programs and data kept for long-term storage or when not in immediate use.

Auxiliary memory may also refer to as auxiliary storage, secondary storage, secondary memory, external storage or external memory. Auxiliary memory is not directly accessible by the CPU; instead, it stores noncritical system data like large data files, documents, programs and other back up information that supplied to primary memory from auxiliary memory over a high-bandwidth channel, which will use whenever necessary. Auxiliary memory holds data for future use, and that retains information even the power fails.

A magnetic disk is a storage device that uses a magnetization process to write, rewrite and access data. It is covered with a magnetic coating and stores data in the form of tracks, spots and sectors. Hard disks, zip disks and floppy disks are common examples of magnetic disks. A magnetic disk primarily consists of a rotating magnetic surface and a mechanical arm that moves over it. The mechanical arm is used to read from and write to the disk. The data on a magnetic disk is read and written using a magnetization process. Data is organized on the disk in the form of tracks and sectors, where tracks are the circular divisions of the disk. Tracks are further divided into sectors that contain blocks of data. All read and write operations on the magnetic disk are performed on the sectors.

Magnetic tape is one of the oldest technologies for electronic data storage. Tape has largely been displaced as a primary and backup storage medium, but it remains well-suited for archiving because of its high capacity, low cost and long durability. It is a linear recording system that is not good for random access. If the tape is part of a library, robotic selection and loading of the right cartridge into a tape drive adds more latency. In an archive, such latencies are not an issue. With tape archiving, there is no online copy for quick retrieval, as everything is vaulted for the long term. While tape can't compete with other media in terms of random access, there are still industries where magnetic tape storage is the preferred technology:

1. Many motion picture production companies record their shoots to tape after experiencing costly failures with both disk and flash.
2. Scientific experiments that produce mass quantities of data in a few microseconds leverage tape's capacity and write speeds.
3. The oil and gas industry has used tape for years to capture, transport and store valuable data. Because oil exploration occurs outside the data center, tape is a good medium for transporting data back from the field.

Tape is often paired with object storage to address the need for lower-latency file access. Sometimes, it is entirely replaced by object storage.

How magnetic tape works

Data bits -- magnetic states representing on and off -- are recorded to a particulate medium bonded to a substrate of Mylar plastic. Improvements in track-following technology and giant magnet resistive read/write heads have increased the number of tracks that can be recorded on a tape.

CACHE MEMORY

Analysis of a 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 localized areas in memory. The phenomenon is known as the property of locality of reference. The reason for this property may be understood considering that a typical computer program flows in a straight-line fashion with program loops and subroutine calls encountered frequently. When a program loop is executed, the CPU repeatedly refers to the set of instructions in memory that constitute the loop. Every time a given subroutine is called, its set of instructions is fetched

from memory. Thus loops and subroutines tend to localize the references to memory for fetching instructions. To a lesser degree, memory references to data also tend to be localized. Table-lookup procedures repeatedly refer to that portion in memory where the table is stored. Iterative procedures refer to common memory locations 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 frequently. 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. Such a fast small memory is referred to as a cache memory. It is placed between the CPU and main memory as illustrated in below Fig. The cache memory access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components. The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory, the average memory access time will approach the access time of the 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 the 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 cache so that future references to memory find the required words in the fast cache memory.

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. 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. 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 ratios 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 the fast cache memory. For example, a computer with cache access time of 100 ns, a main memory access time of 1000 ns, and a hit ratio of 0.9 produces an average access time of 200 ns. This is a considerable improvement over a similar computer without a cache memory, whose access time is 1000 ns. The basic characteristic of cache memory is its fast access time. Therefore, very little or no time must be wasted when searching for words in the cache. The transformation of data from main memory to cache memory is referred to as a mapping process. Three 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

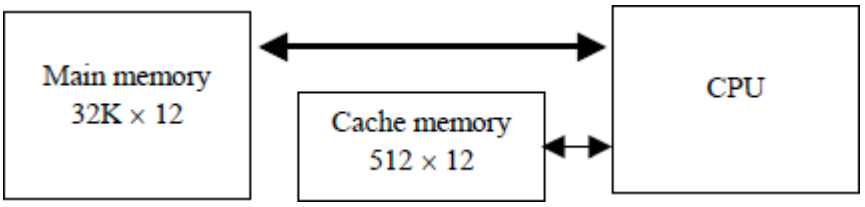


Figure - Example of cache memory

ASSOCIATIVE MEMORY

Many data-processing applications require the search of items in a table stored in memory. An assembler program searches the symbol address table in order to extract the symbol's binary equivalent. An account number may be searched in a file to determine the holder's name and account status. The established way to search a table is to store all items where they can be addressed in sequence. The search procedure is a strategy for choosing a sequence of addresses, reading the content of memory at each address, and comparing the information

read with the item being searched until a match occurs. The number of accesses to memory depends on the location of the item and the efficiency of the search algorithm. Many search algorithms have been developed to minimize the number of accesses while searching for an item in a random or sequential access memory. The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address. A memory unit accessed by content is called an associative memory or content addressable memory (CAM). This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location. When a word is written in an associative memory, no address is given,. The memory is capable of finding an empty unused location to store the word. When a word is to be read from an associative memory, the content of the word, or part of the word, is specified. The memory locaters all words which match the specified content and marks them for reading. Because of its organization, the associative memory is uniquely suited to do parallel searches by data association. Moreover, searches can be done on an entire word or on a specific field within a word. An associative memory is more expensive then a random access memory because each cell must have storage capability as well as logic circuits for matching its content with an external argument. For this reason, associative memories are used in applications where the search time is very critical and must be very short.

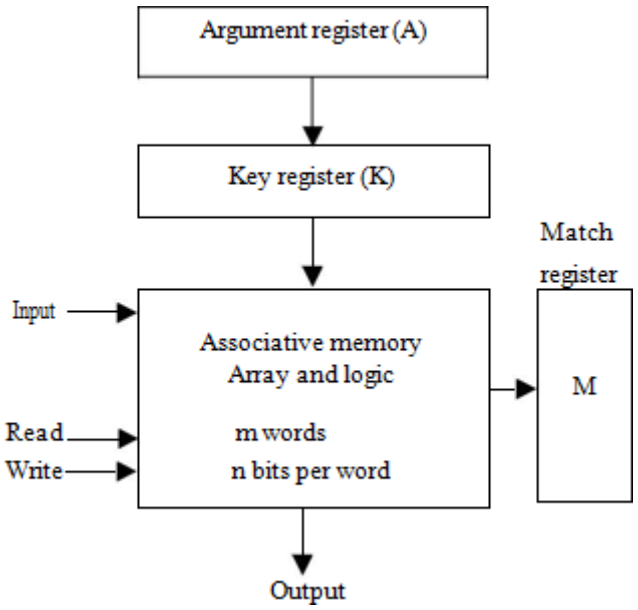
HARDWARE ORGANIZATION

The block diagram of an associative memory is shown in below Fig. It consists of a memory array and logic from words with n bits per word. The argument register A and key register K each have n bits, one for each bit of a word. The match register M has m bits, one for each memory word. Each word in memory is compared in parallel with the content of the argument register. The words that match the bits of the argument register set a corresponding bit in the match register. After the matching process, those bits in the match register that have been set indicate the fact that their corresponding words have been matched. Reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.

The key register provides a mask for choosing a particular field or key in the argument word. The entire argument is compared with each memory word if the key register contains all 1's. Otherwise, only those bits in the argument that have 1's in their corresponding position of the

key register are compared. Thus the key provides a mask or identifying piece of information which specifies how the reference to memory is made.

Figure- Block diagram of associative memory



To illustrate with a numerical example, suppose that the argument register A and the key register K have the bit configuration shown below. Only the three leftmost bits of A are compared with memory words because K has 1's in these positions.

A 101 111100

K 111 000000

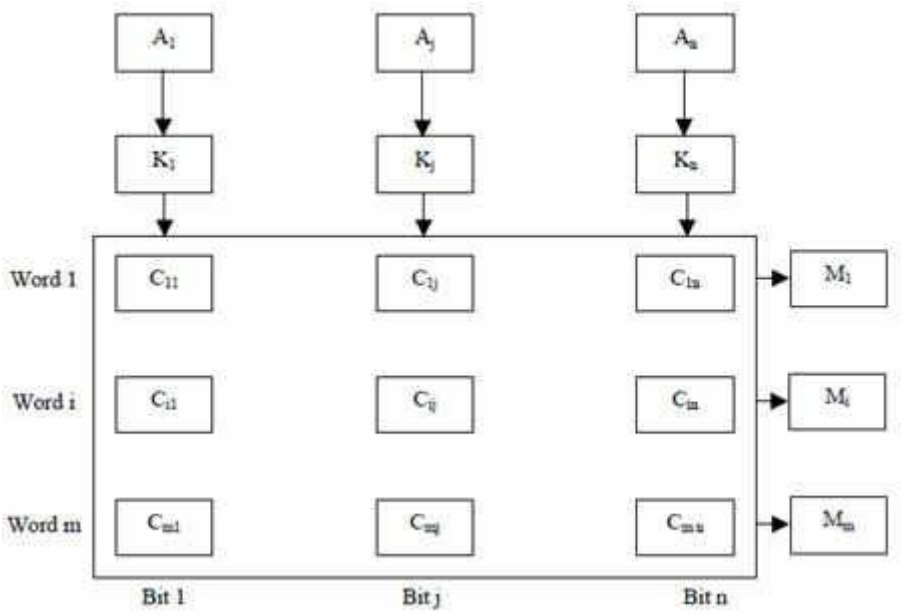
Word 1 100 111100 no match

Word 2 101 000001 match

Word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal. The relation between the memory array and external registers in an associative memory is shown in below Fig. The cells in the array are marked by the letter C with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. Thus cell Cij is the cell for bit j in word i. A bit A j in the argument register is compared with all the bits in column j of the array provided that K j = 1. This is done for all columns j = 1, 2,…,n. If a match occurs between all the unmasked bits of the

argument and the bits in word i, the corresponding bit Mi in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, Mi is cleared to 0.

Figure -Associative memory of m word, n cells per word



It consists of a flip- Flop storage element Fij and the circuits for reading, writing, and matching the cell. The input bit is transferred into the storage cell during a write operation. The bit stored is read out during a read operation. The match logic compares the content of the storage cell with the corresponding unmasked bit of the argument and provides an output for the decision logic that sets the bit in Mi.

Memory Management The task of the memory manager and memory management are to ensure that all processes are always able to access their memory. To accomplish this task requires careful integration between the computer's hardware and the operating system. When several processes with dynamic memory needs run on the computer at the same time, it is necessary to reference data with both a logical address and a physical address. The hardware is responsible for translating the logical addresses into physical addresses in real time. While the operating system is responsible to:

1. ensure that the requested data is in physical memory when needed
2. Program the hardware to perform the address translations.

The Paged Memory Management scheme gives rise to the notion of demand paging using virtual memory. The Virtual Memory Management system maintains a copy of the memory for all programs on secondary storage, such as a hard drive. In fact, many pages for a process may only reside in virtual memory. Loading only the page frames that are needed to run a program can make it faster to load a program. Some memory frames for a program may never be needed while the program runs.

If a current copy of a frame of physical memory is held on disk in virtual memory, then that frame may be removed from physical memory to free up needed memory. When a process references memory that is not loaded in physical memory, the Memory Management Unit of the CPU issues a page fault trap.