Chapter – 1

Introduction

Computer Organization and Computer Architecture:

Computer Organization:

- The organization of a computer refers to the logical structure of the system, including the CPU, control unit, I/O, etc.
- Computer organization is how operational attributes are linked together and contribute to realize the architectural specifications.
- A computer's organization expresses the realization of the architecture.
- Architecture describes what the computer does organization describes how it does it.

Computer Architecture:

- The architecture of a computer is the set of resources seen by the computer programmer. It includes general purpose registers, the status word, the instruction set, the address space, etc.
- Computer architecture is the architectural attributes like physical address memory, CPU and how they should be made and made to coordinate with each other keeping the future demands and goals in mind.
- A computer's architecture is its abstract model and is the programmer's view in terms of instructions, addressing modes and registers.

Review of Evolution of Computer System:

The computer as we know it today had its beginning with a 19th century English mathematics professor name Charles Babbage. He designed the Analytical Engine and it was this design that the basic framework of the computers of today are based on. Generally speaking, computers can be classified into three generations. Each generation lasted for a certain period of time, and each gave us either a new and improved computer or an improvement to the existing computer.

First Generation of Computers:

The computers which were discovered approximately between 1942 A.D. and 1955 A.D. are classified as the First Generation of Computers. All the computers which were discovered during the first generation had "Vacuum Tubes" for their memory and processing devices. Vacuum tube was developed by Lee De Forest in 1908 A.D. and used later in computer system.

Vacuum tubes were a glass device, which used filaments as a source of electronics, could control and amplify electronic signals. It was the only high speed electronic switching device available in those days. The vacuum tubes required great amount of energy and generated much heat. Lots of space were required not only for the large number of vacuum tubes but also for housing special air-conditioning units to get rid of heat generated by vacuum tubes.

Examples: Mark-I, ENIAC, EDSAC, EDVAC, UNIVAC-I, ABC, IBM 701, IBM 700 Series, IBM 704, IBM 709, etc.

Second Generation of Computers:

The computers which were discovered approximately between 1955 A.D. and 1964 A.D. are classified as Second Generation of Computers. They have transistor and diodes for their memory and processing devices. Transistor was developed by three scientists John Braden, William Shockley and Walter Brattain in 1947 A.D. at Bell laboratory in United States and won the Nobel Prize in 1956 A.D. for it.

Transistors were made of solid materials principally called silicon and germanium semiconductor material rather than glass. Therefore they were cheap to produce. One transistor replaced the equivalent of 40 vacuum tubes. Transistors were found to conduct electricity faster and better than vacuum tubes. They were also much smaller and gave off virtually no heat compared to vacuum tubes. Their use marked a new beginning for the computer. Transistors were highly reliable as compared to vacuum tubes, since they had no part like filament, which could burn out. They could switch much faster (almost 10 times faster) than vacuum tubes. Transistor is the basic unit in radio, television and computer circuits. It is often used to amplify the current flowing from one circuit to another. Transistor consists three connecting parts: a base, an emitter and a collector.

Example: IBM 7090, IBM 7094I, IBM 7094II, IBM 1620, IBM 1401 (first computer brought in Nepal on hire for National Census 2028), ICL 2950/10 (second computer brought in Nepal from England on 20 lakhs US dollar for National Census 2038), etc.

Third Generation of Computers:

The computers which were discovered approximately between 1964 A.D. and 1975 A.D. are classified as Third Generation of Computers. They have Integrated Circuits for memory and processing devices. Integrated Circuit was developed by Jack St. Clair Kilby and Robert Noyce in 1958 A.D.

An integrated Circuit (IC) also called a microchip, is an electronic circuit consisting of a large number of electronic components like transistors, resistors and capacitors placed on a single silicon chip, eliminating wired interconnection between components. IC chips are much smaller in size, faster in operation, consumed much less power, high performance and more reliable than transistor and vacuum tubes. Initially, the integrated circuits contained only up to 100 components and the technology named as SSI (Small Scale Integration). Later, with the advancement in technology for manufacturing ICs, it became possible to integrate from 100 to 3000 components on a single chip and this technology named as MSI (Medium Scale Integration). They function as timers, amplifiers, logic units, counters, calculators, temperature sensors, and radio receivers.

Example: IBM system/360, ICL 1900, IBM 370 series, Honeywell 2200 series, CDC 7600, STAR-100, UNIVAC 9000, etc.

Fourth Generation of Computers:

The computers which were discovered approximately between 1975 A.D. and 1989 A.D. are classified as Fourth Generation of Computers. They have microprocessor as CPU "Central

Processing Unit" with LSI "Large Scale Integration" and VLSI "Very Large Scale Integration" technology in ICs as memory and processing devices. Microprocessor was developed by Intel Corporation in 1971 A.D.

A microprocessor is a single chip in which millions of components like transistors are integrated together in different layers and it performs all the operations of the computers processor; since it is so small, it is called a microprocessor. It is a complete CPU built on a single chip by using LSI or VLSI technology.

Intel Corporation of USA developed the first microprocessor named "Intel 4004" in 1971 A.D. It contained about 1600 transistors. It was a 4-bit microprocessor and process only 4 bits of data at a time. Since then, the technology has increased by leaps and bounds. The microprocessors available today are more powerful than many of the large computers of the past. They have become cheaper and more reliable too. The modern processors are available in 32, 64 or higher bits word length. The important characteristics of a microprocessor are the width of address bus, data bus, clock speed and its instruction set architecture. Processors are also often classified as being either RISC "Reduced Instruction Set Computer" or CISC "Complex Instruction Set Computer".

Example: IBM PC, Apple/Macintosh, Wang Laser, Letron, Acer ASPIRE 5741, Apple MacBook Air, Dell Inspiron 1440, etc.

Fifth Generation of Computers:

The computers which were discovered approximately between 1989 A.D. and present are classified as Fifth Generation of Computers. In the fifth generation, VLSI "Very Large Scale Integration" technology became ULSI "Ultra Large Scale Integration" technology, resulting in the production of microprocessor chips having ten million electronic components. In fact, the speed of microprocessors and the size of main memory and hard disk doubled almost every 18 months.

This generation is based on parallel processing hardware and Artificial Intelligence "AI" software. Artificial Intelligence is an emerging branch in computer science, which interprets means and method of making computers think like human beings. All the high level languages like C and C++, Java, .Net, etc are used in this generation.

During fifth generation, there was tremendous outgrowth of computer networks which emerge the use of electronic mail and World Wide Web "WWW". Moreover, exciting applications like electronic commerce, virtual libraries, virtual classrooms, distance education, etc emerged during the period.

Generally this generation computers are referred as Artificial Intelligence which includes:

- Robotics
- Neural Networks
- Game playing
- Expert Systems
- Understand natural languages such as Nepali, English, Hindi, etc.

As a result of the various improvements to the development of the computer we have seen the computer being used in all areas of life. It is a very useful tool that will continue to experience new development as time passes.

BASIC STRUCTURE OF COMPUTER SYSTEM:

Structure and Function:

Computer is a complex system as it contains millions of electronic components. It simply refers to the architecture of a computer system. The key concept if to recognize hierarchical model of more complex system and each complex system is divided to lower level hierarchical model until we reach lowest level of elementary sub-system.

The structure refers to the way in which components are inter-related. Similarly, function refers to the operation of each individual component as a part of structure. The functional view of computer system is shown below:

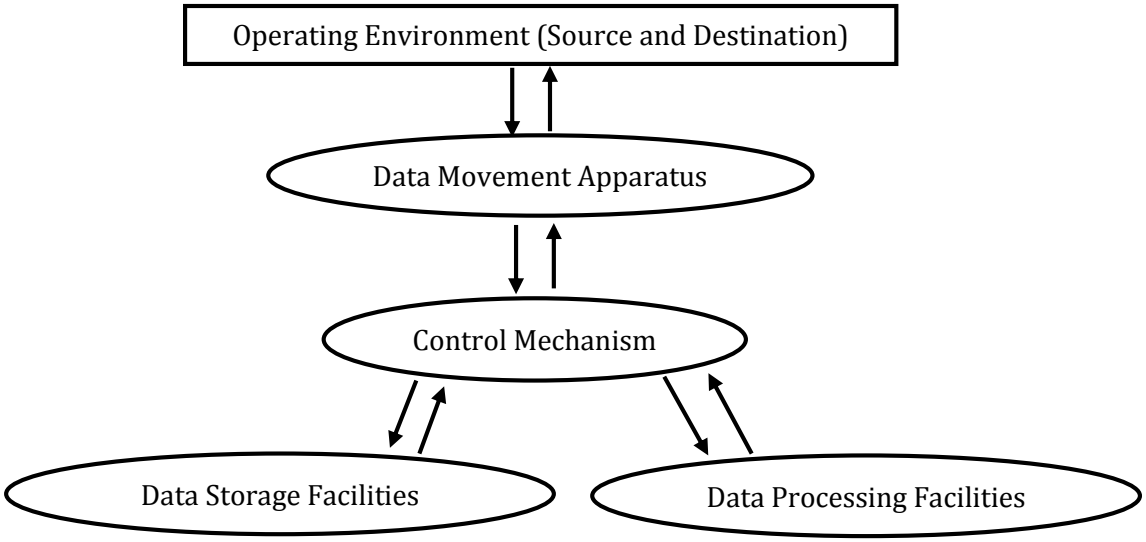


Fig: Functional View of Computer System

It is essential that computer stores data even if computer is processing data on the flying mode *(data coming, get process and result go out)*. A computer must temporarily store at least those pieces of data that are being work at any given movement. Thus, there is at least a short term data storage function.

Except it computer uses different hierarchy of memory like primary memory, secondary memory, etc. The computer must be able to move data between itself and outside world. The computer operating environment consists of devices that serves as either source or destination whereas control mechanism controls all the activities.

Structure of Computer System (Top Level):

The computer system has mainly four parts and they are:

1. **CPU:** It controls the operation of computer and perform its data processing.
2. **Main Memory:** It stores the data.
3. **Input/output:** They moves data between computer and external environment.
4. **System Interconnection:** Some mechanism that provides communication among CPU, Main Memory and Input/output.

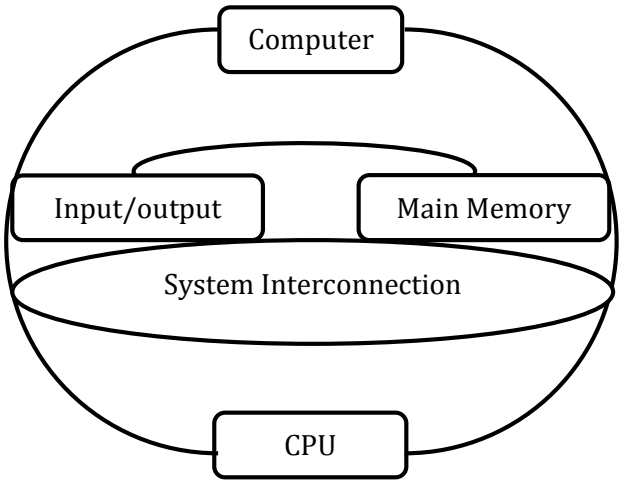


Fig: Structure of Computer System (Top Level)

Structure of CPU:

Some major components of CPU structure are:

1. **Control Unit:** It controls overall operations of CPU.
2. **ALU:** It performs arithmetic and logical operations.
3. **Registers:** It provides storage inside the CPU.
4. **CPU Interconnection:** It provides communication between register, ALU and Control Unit.

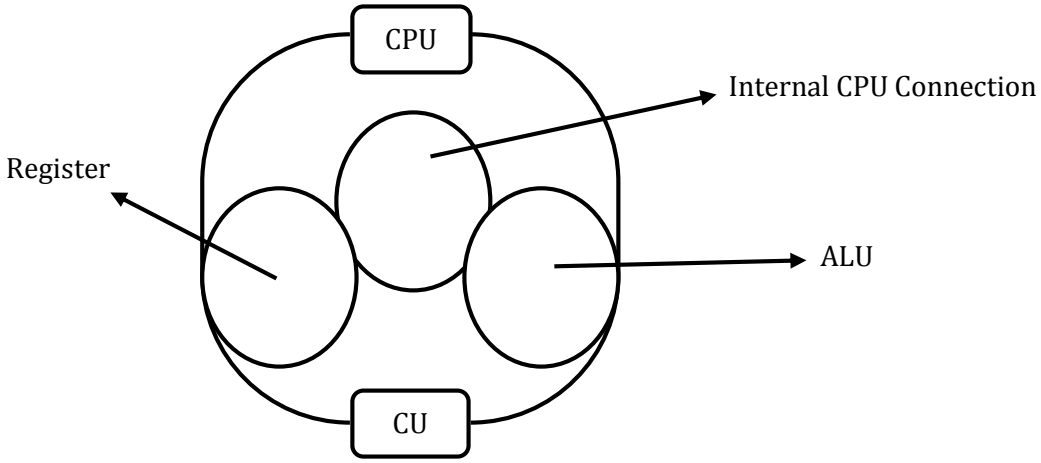


Fig: Structure of CPU

Structure of Control unit:

The structure of control unit contains following components:

1. **Sequence Logic:** Digital circuit whose output depends on current input and state of circuits.
2. **Control Memory:** The register which stores data related to controlled operation.

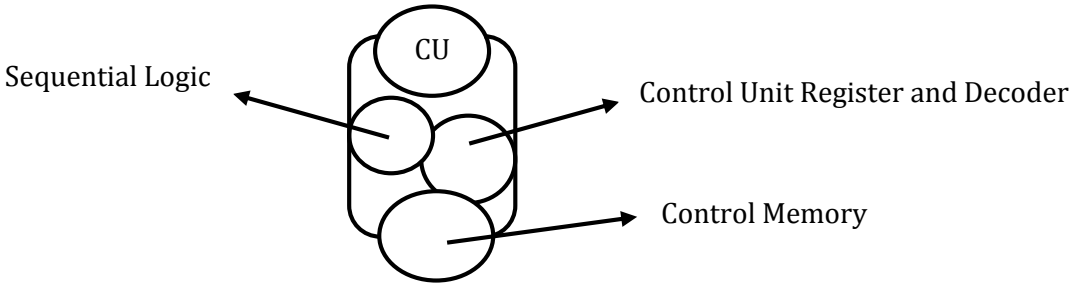


Fig: Structure of Control Unit

EXAMPLES OF COMPUTER FAMILIES:

Pentium is the family of microprocessor developed by Intel Corporation which was introduced in 1993 as a successor of Intel 80486 microprocessor. It was developed by CISC architecture (Complex Instruction Set Computer). Its main feature was 32 bit address bus memory management unit. The evolution of Intel Pentium processor is described below:

1. **4-bit Processor:** Intel 4004 was the first microprocessor originally developed to be used in calculator.
2. **8-bit Processor:** Intel 8008 was the first 8-bit microprocessor originally intended for use in data point.
3. **16-bit Processor:** It has data bus for 19-bit and address bus of 20-bit. It includes 8086, 80186 and 80286 microprocessors.
4. **32-bit Processor:** It includes 80386 and 80486 processor which supports multitasking.
5. **Pentium Series:** with the Pentium, Intel introduced the use of superscalar technology which allow multi-instructions to execute in parallel. The Pentium series are:
   - Pentium Pro
   - Pentium II
   - Pentium III
   - Pentium IV

FUTURE TRENDS IN COMPUTER:

1. **5G:**

Enabling everything from interactive automobiles and super gaming to the industrial Internet of Things, 5G will take wireless to the future and beyond, preparing for the rapidly approaching day when everything, **including** the kitchen sink, might be connected to a network, both local and the Internet.

2. **Virtual Reality**:

VR technologies reach a critical mass of functionality, reliability, ease of use, affordability and availability. Movie studios are partnering with VR vendors to bring content to market. News organizations are similarly working with VR companies to bring immersive experiences of news directly into the home, including live events

3. **Nonvolatile Memory:**

Nonvolatile memory, which is computer memory that retrieves information even after being turned off and back on, has been used for secondary storage due to issues of cost, performance and write endurance, as compared to volatile RAM memory that has been used as primary storage. This will literally change the landscape of computing, allowing smaller devices to store more data and large devices to store huge amounts of information.

4. **Cyber Physical Systems (CPS**):

CPS are smart systems that have cyber technologies, both hardware and software, deeply embedded in and interacting with physical components, sensing and changing the state of the real world. These systems have to operate with high levels of reliability, safety, security and usability since they must meet the rapidly growing demand for applications such as the smart grid, the next generation air transportation system, intelligent transportation systems, smart medical technologies, smart buildings and smart manufacturing.

5. **Data Science:**

Technically, data science is an interdisciplinary field about processes and systems to extract knowledge or insights from data in various forms, either structured or unstructured, which is a continuation of some of the data analysis fields such as statistics, data mining and predictive analytics. In less technical terms, a data scientist is an individual with the curiosity and training to extract meaning from big data, determining trends, buying insights, connections, patterns and more. Frequently, data scientists are mathematics and statistics experts. Sometimes, they're more generalists; other times, they are software engineers.

6. **Capability-Based Security:**

The greatest single problem of every company and virtually every individual in this cyber world is security. The number of hacks rises exponentially every year, and no one's data is safe. Finding a "better way" in the security world is golden. Hardware capability-based security, while hardly a household name, may be a significant weapon in the security arsenal of programmers, providing more data security for everyone. Capability-based security will provide a finer grain protection and defend against many of the attacks that today are successful.

7. **Advanced Machine Learning:**

Impacting everything from game playing and online advertising to brain/machine interfaces and medical diagnosis, machine learning explores the construction of algorithms that can learn from and make predictions on data. Rather than following strict program guidelines, machine learning systems build a model based on examples and then make predictions and decisions based on data. They "learn."

8. **Network Function Virtualization (NFV):**

More and more, the world depends on cloud services. Due to limitations in technology security, these services have not been widely provided by telecommunications companies — which is a loss for the consumer. NFV is an emerging technology which provides a virtualized infrastructure on which next-generation cloud services depend. With NFV, cloud services will be

provided to users at a greatly reduced price, with greater convenience and reliability by telecommunications companies with their standard communication services.

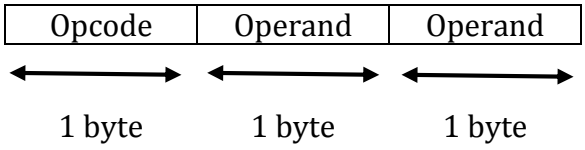
9. **Containers:**

Containers allow companies to develop and deliver applications faster and more efficiently. This is a boon to consumers, who want their apps fast. Containers provide the necessary computing resources to run an application as if it is the only application running in the operating. While containers can deliver many benefits, the gating item is security, which must be improved to make the promise of containers a reality.

INSTRUCTION AND INSTRUCTION FORMAT:

An instruction is a command given to the computer to perform a specific task.

Instructions are represented as a sequence of bits. An instruction is divided in to number of fields corresponding to elements of instruction. A layout of instruction is known as instruction format. For example: if an instruction is of 3 bytes then its layout is represented as:



INSTRUCTION SETS DESIGN ISSUES:

An instruction set is a complex job because it affects so many aspects of computer system. The instruction set defines many of the functions performed by CPU and therefore, has a significant effect on the implementation of the CPU. Some instruction sets are the means by which a programmer can control CPU. Therefore programmer view must be considered while designing instruction set.

Some of the important issues related to instruction design are:

1. **Operation:** How many and which operation to provide and how complex operation should be performed.
2. **Data Types:** The variable types of data on which the operations are performed.
3. **Registers:** The number of registers that can be referenced to operand is specified.
4. **Registered:** This include instruction length number of address size, etc.

MILESTONE IN COMPUTER ORGANIZATION:

In development process of computer new concept is invented and used. One of the milestone concept is stored program concept (SPC). In this concept of computer design instructions are stored in memory and are used by accessing that memory by a processor.

In 1946, Von-Neumann and his colleagues design a new stored programmed computer called IAS (Institute for Advanced Study) computer.

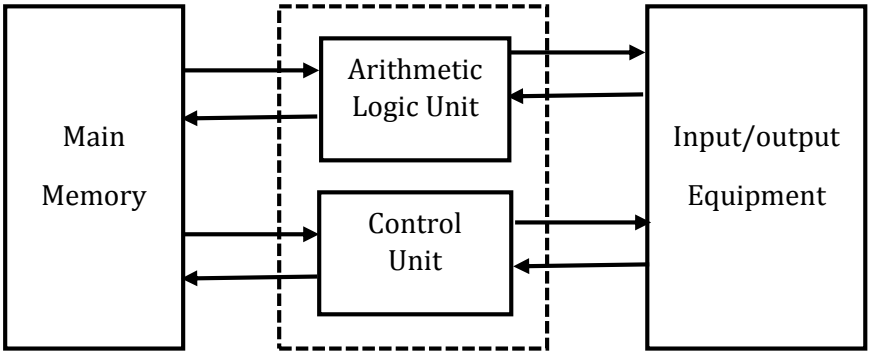
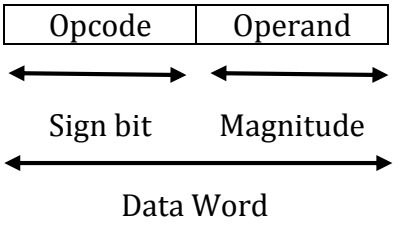


Fig: General Structure of IAS Computer

INSTRUCTION FORMAT OF IAS COMPUTER:

IAS machine consists of thousands of memory location. The location is word and have 40-bit length.



- A number or data word contains 1-bit sign and remaining 39-bit for magnitude.
- The address value may be 0 – 999

EXPANDED STRUCTURE OF IAS COMPUTER:

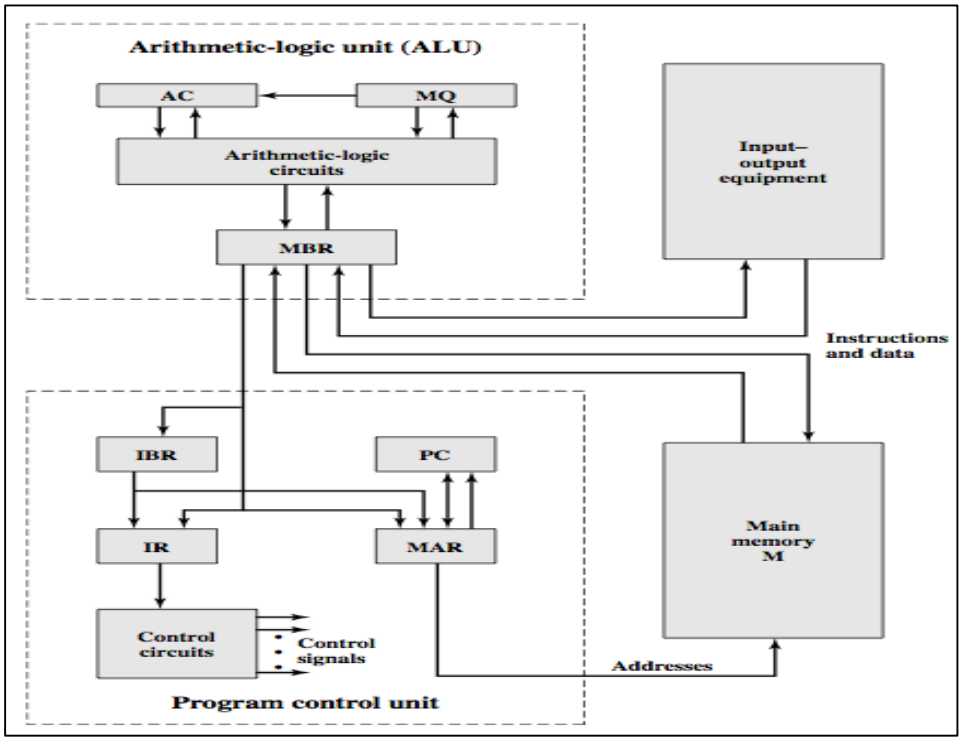


Fig: Expanded Structure of IAS Computer

Before executing an instruction the address, data and instructions are stored in register. The CPU consisting Program Control Unit and Arithmetic Logic Unit have small registers which are as follows:

1. **Memory Buffer Register (MBR):** It contains a word to be stored in memory or is used to receive a word from memory.
2. **Memory Address Register (MAR):** It contains the address value to specific address in memory of the word to be written from or read into MAR.
3. **Instruction Register (IR):** It controls Opcode of the instruction begin executed.
4. **Instruction Buffer Register (IBR):** It temporarily holds an instruction from a word in memory.
5. **Program Counter (PC):** It contains the address of next instruction to be fetched from memory.
6. **Accumulator and Multiplier Quotient (AC and MQ):** Holds temporary operands and result of ALU operation.

IAS INSTRUCTIONS:

1. Data Transfer Group:

Move data between memory and ALU register and vice-versa.

a. **Load MQ:** Transfer contents of register MQ to accumulator.
b. **Load MQ, M(X):** Transfer contents of memory location X to MQ.
c. **Store M(X):** Transfer content of accumulator.
d. **Load M(X):** Transfer M(X) to accumulator.
e. **Load –M(X):** Transfer –M(X) to accumulator.

2. Arithmetic Instruction:

These are the instructions performed by ALU.

a. **Add M(X):** Add M(X) to accumulator.
b. **Sub M(X):** Subtract M(X) from accumulator.
c. **MUL M(X):** Multiply M(X) and MQ.
d. **DIV M(X):** Divide accumulator by M(X).

3. Unconditional Branching Instructions:

The sequence of execution of instruction changes by unconditional branch. The purpose is to execute instructions repeatedly.

a. **Jump M(X; 0:19):** Take next instruction from left half of M(X).
b. **Jump M(X; 20:39):** Take next instruction form right of M(X).

4. Conditional Branch Instruction:

The jump depends on condition.

a. **Jump +M(X; 0:19):** If the number in accumulator is non-negative take next instruction from left half of M(X).

b. **Jump –M(X; 20:39):** If the number in accumulator is negative take next instruction from right half of M(X).

5. Address Modify:

Permits address to be compared by ALU and then inserted in instruction stored in memory.

a. **STOR M(X; 8:19):** Replace left address field at M(X) by 12 right most bits of accumulator.
b. **STOR M(X; 28:39):** Replace right address filed at M(X) by 12 right most bits of accumulator.

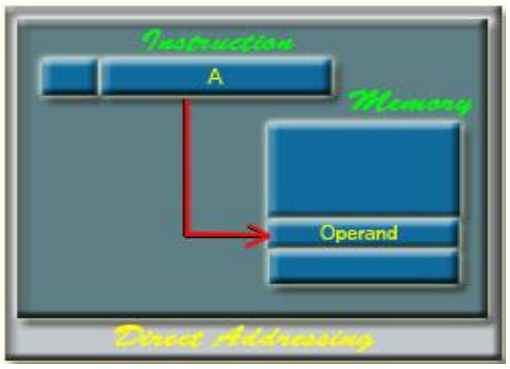
Addressing Mode:

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in computer register as memory words. The way the operands are chosen during program execution is dependent on the addressing mode of the instruction. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction between the operand is activity referenced. Computer use addressing mode technique for the purpose of accommodating one or both of the following provisions.

1. To give programming versatility to the uses by providing such facilities as pointer to memory, counters for top control, indexing of data, and program relocation.
2. To reduce the number of bits in the addressing fields of the instruction.

The most common addressing techniques are:

1. Direct Addressing:

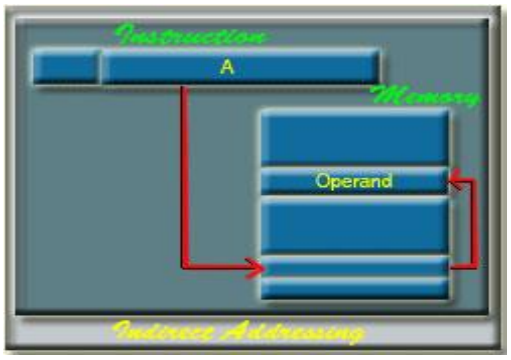


A very simple form of addressing is direct addressing, in which the address field contains the effective address of the operand:

EA = A

It requires only one memory reference and no special calculation.

2. Indirect Addressing:



With direct addressing, the length of the address field is usually less than the word length, thus limiting the address range. One solution is to have the address field refer to the address of a word in memory, which in turn contains a full-length address of the operand. This is known as indirect addressing:

EA = (A)

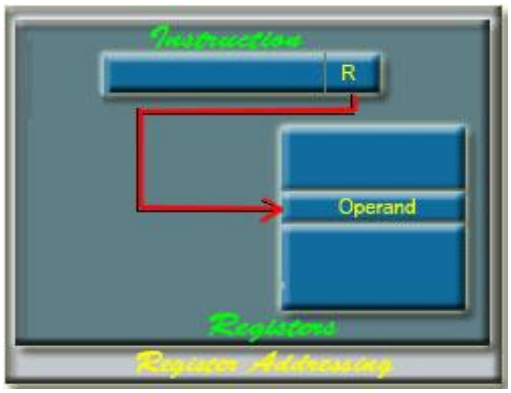
3. Immediate Addressing:

The simplest form of addressing is immediate addressing, in which the operand is actually present in the instruction:

OPERAND = A

This mode can be used to define and use constants or set initial values of variables. The advantage of immediate addressing is that no memory reference other than the instruction fetch is required to obtain the operand. The disadvantage is that the size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the world length.

4. Register Addressing:

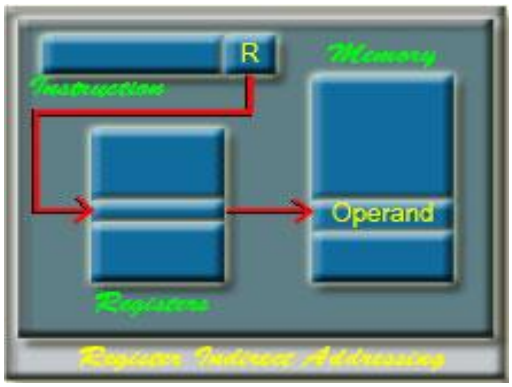


Register addressing is similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address:

EA = R

The advantages of register addressing are that only a small address field is needed in the instruction and no memory reference is required. The disadvantage of register addressing is that the address space is very limited.

5. Register Indirect Addressing:



Register indirect addressing is similar to indirect addressing, except that the address field refers to a register instead of a memory location. It requires only one memory reference and no special calculation.

EA = (R)

Register indirect addressing uses one less memory reference than indirect addressing. Because, the first information is available in a register which is nothing but a memory address. From that memory location, we use to get the data or information. In general, register access is much faster than the memory access.

6. Relative Addressing:

For relative addressing, the implicitly referenced register is the program counter (PC). That is, the current instruction address is added to the address field to produce the EA. Thus, the effective address is a displacement relative to the address of the instruction.

7. Base-Register Addressing:

The reference register contains a memory address, and the address field contains a displacement from that address. The register reference may be explicit or implicit. In some implementation, a single segment/base register is employed and is used implicitly. In others, the programmer may choose a register to hold the base address of a segment, and the instruction must reference it explicitly.

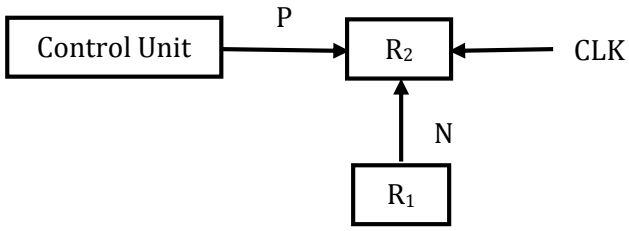
Chapter – 2

Register Transfer and Micro-operations

Register Transfer:

As we know that the digital computer has many registers and while performing certain tasks the content of one register may change or transfer to another. The operations or micro-operation which transfers content of one register to another is known as register transfer.

The register transfer is represented as: $R_2 \leftarrow R_1$. It implies that the content of register R_1 is transfer to R_2. The control signal can also be known as P: $R_2 \leftarrow R_1$. The important hardware implementation is:



The data transfer (Register to Register) occurs in a single clock pulse known as T-state. A T-state is the smallest unit of time in which smallest operation is performing.

Register Transfer Language (RTL):

It is a symbolic notation used to describe the operation in a register transfer, which describes the elementary operation of digital computer system. Language is totally computer architecture dependent and hence RTL defines symbol or various types of micro-operations.

Micro-operation is a primitive action performed by a machine on the data stored in one or more-registers. We can say that micro-operations are functional or automatic operation of processor.

Types of Micro-operations:

1. Register Transfer Micro-operation:

These micro-operations transfer information from one register to another. The information contained in the register does not change during micro-operation. It can written as: **$R_2 \leftarrow R_1$**. Here, the content of R_1 is transfer to R_2 and for this operation there must be a data path for data transfer from source register to destination register.

2. Arithmetic Micro-operation:

These micro-operations performs on arithmetic operations on numeric data stored in register. The basic operation may be addition, subtraction, increment, decrement, etc.

Example:

R3 ← R1+R2
R2 ← R1- R3
R1 ← R2 + 1
R3 ← R1 – 1

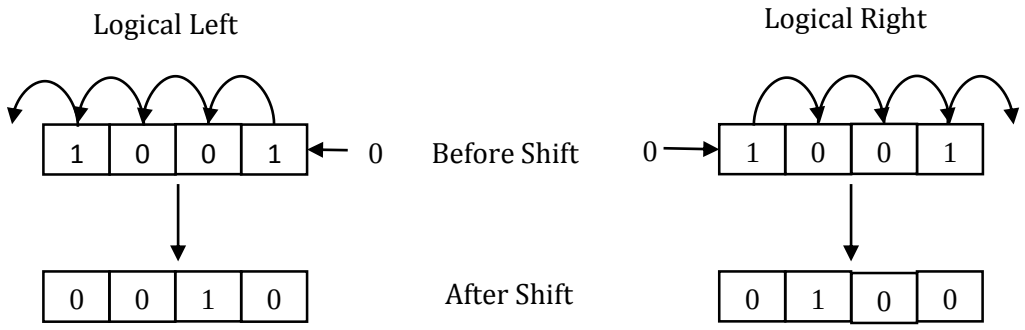
3. Logical Micro-operation:

They basically perform binary operation on the bits of a string stored in a register. For a logical micro-operation each bit of a register is considered as a variable. Some of the common logical micro-operations are: AND, OR, NAND, NOT, etc.

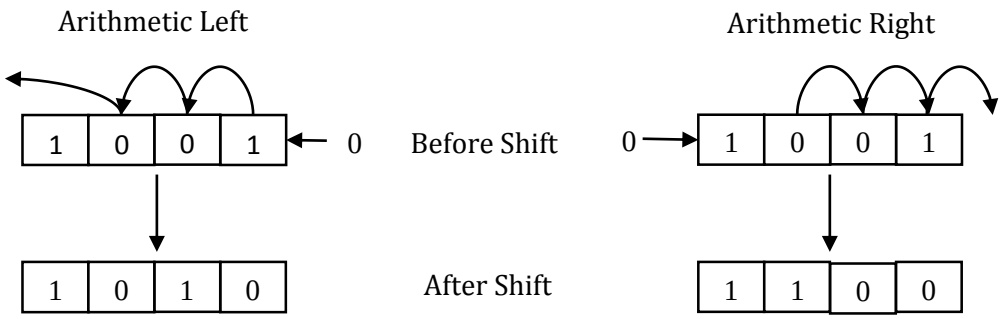
4. Shift Micro-operation:

Shift is a useful operation which can be shifted as a serial data transfer and they can also be along with arithmetic and logical operations.

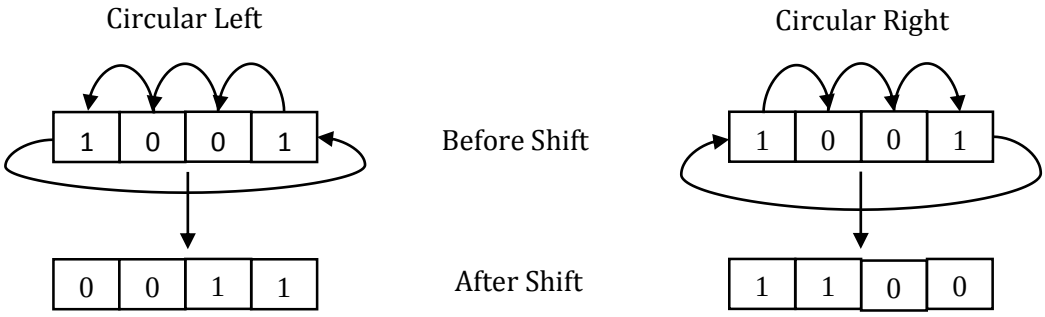
a. Logical Shift:



b. Arithmetic Shift (Sign Bit Should Not Change):



c. Circular Shift:



INTRODUCTION TO HDL:

In electronics, a hardware description language (HDL) is a specialized computer language used to describe the structure and behavior of electronic circuits, and most commonly, digital logic circuits.

A hardware description language enables a precise, formal description of an electronic circuit that allows for the automated analysis and simulation of an electronic circuit. It also allows for the synthesis of a HDL description into a netlist (a specification of physical electronic components and how they are connected together), which can then be placed and routed to produce the set of masks used to create an integrated circuit.

A hardware description language looks much like a programming language such as C; it is a textual description consisting of expressions, statements and control structures. One important difference between most programming languages and HDLs is that HDLs explicitly include the notion of time.

INTRODUCTION TO VHDL:

VHDL stands for very high-speed integrated circuit hardware description language. Which is one of the programming language used to model a digital system by dataflow, behavioral and structural style of modeling. This language was first introduced in 1981 for the department of Defense (DoD) under the VHSIC programe. In 1983 IBM, Texas instruments and Intermetrics started to develop this language. In 1985 VHDL 7.2 version was released. In 1987 IEEE standardized the language.

This language is hardware dependent, where any digital system can be simulated or can be represented in VHDL. This language is only used in digital system. Other languages such as ABEL (Advanced Boolean Expression Language) and VEILOG are also used for hardware description but among these VHDL is more popular and widely used.

The program structure for VHDL is:

```
entity-name is;
[PORT] (interface-signal-declaration);
end (entity) entity-name;
```

architecture architecture-name of entity is;
[declaration]
begin;
architecture-body;
end [architecture] architecture-name;

Example:

NOR = OR + NOT
Entity-NOR is;
[PORT] (input A, B; Output Y)
End NOR;
Architecture NOR of NOR is;
Input A, B; X-output;
Input X; Y-output;
Begin;
A, B; Y
End NOR

Chapter – 3

Central Processing Unit

Computer Organization/Structure:

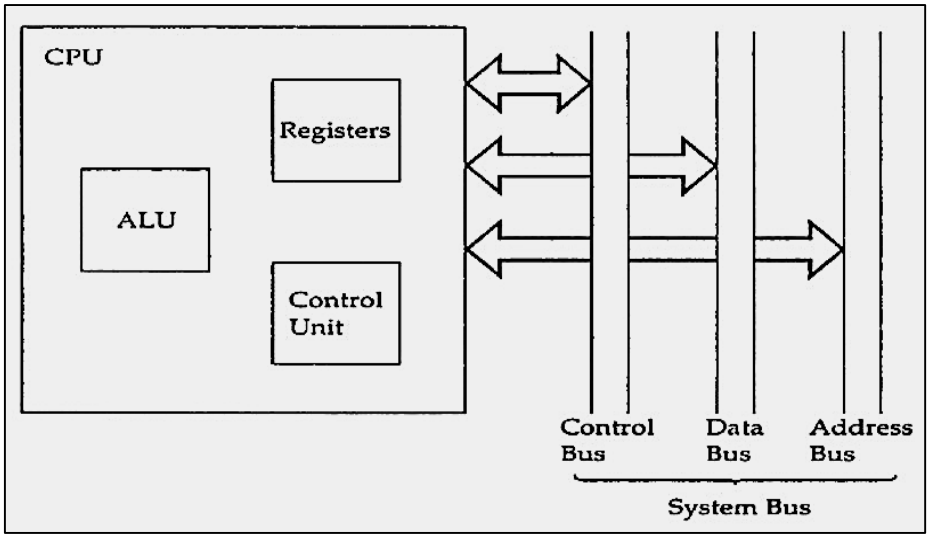


Fig: General View of CPU

CPU consists of three components that is ALU, CU and Register, where ALU performs Arithmetic and Logical operation, CU provides control signals and Registers store binary information which provides fast information to ALU.

These all components are connected with each other and with external environment through system bus as shown in figure above. The operations that must be performed by CPU are as follows:

1. **Fetch Instruction:** The CPU reads an information form memory.
2. **Interpret Instruction:** The instruction is decoded to determine what action is required.
3. **Fetch Data:** The execution of an instruction may require reading data from memory or an input/output module.
4. **Process Data:** The execution of an instruction may require performing some arithmetic or logical operations on data.
5. **Write Data:** The result of an execution may require writing data to memory or an input/output module.

Detailed View of CPU:

The figure shown below is detailed view of CPU. The data transfer and logic control path are indicated including an element labelled as internal CPU BUS. The different parts of ALU are:

1. **Status Flag:** It stores status of any operation like zero, sign, overflow, etc.
2. **Shifter:** Performs left and right shift.
3. **Complementer:** Performs bitwise operation.

4. **Arithmetic and Boolean Logic:** Performs arithmetic and logical operations.
5. **Registers:** Collection of all registers like IR, MBR, MAR, etc.

Similarly, control unit is another important part of CPU organization, which provides control signals to all components like READ, WRITE, HALT, etc.

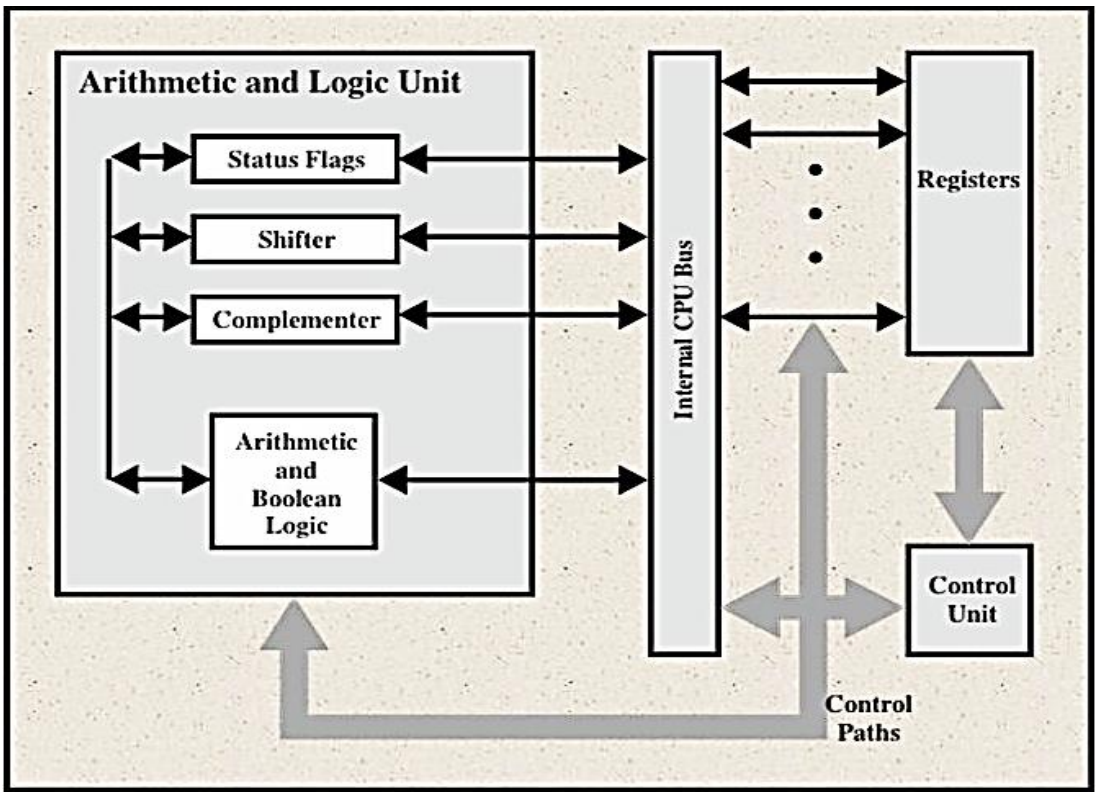


Fig: Detailed View of CPU

REGISTER ORGANIZATION:

Within CPU, there is a set of register that works as an internal memory of CPU. The number of register set in different computer may differ. All these registers can be categorized into two groups which are as follows:

1. USER/PROGRAMMABLE:

These registers can be used by machine or assembly language programmer to minimize the references to main memory. In general there are four types of User visible registers:

a. General Purpose Registers (GpRs):

They are used for various functions desired by processor. It may contain an operand or can be used for calculation of address of operand and can even store some data.

b. Data Register:

They are used only for storing immediate result or data. These data registers are not used for address calculation of an operand.

c. **Address Registers:**

It may be any general purpose register but in some cases few dedicated address registers are used such as:

- **Segment Pointer:** Used to point out segment of memory.
- **Index Register:** Used for index addressing scheme.
- **Stack Pointer:** Used for storing address of top of stack.

d. **Flag Register:**

They are also known as conditional code registers. They are used for set or reset the condition of operation.

2. Status and Control Registers:

For control of various operation several registers are used. These registers cannot be used in data manipulation. Some of the important status and control registers are:

a. **Program Counter:** Contain address of next instruction
b. **Instruction Register:** Contain instruction most recently fetched.
c. **Memory Address Register:** Contain address of instructions and data.
d. **Memory Buffer Register:** Contains a word of data to be written to memory or data word that are most recently used.

DATA PATHS:

A datapath is a collection of functional units (such as arithmetic logic units or multipliers, which perform data processing operations), registers, and buses. Along with the control unit it composes the central processing unit (CPU).

During the late 1990s, there was growing research in the area of reconfigurable datapaths, which may be re-purposed at run-time using programmable fabric, as such designs may allow for more efficient processing as well as substantial power savings.

Functional Blocks of a Datapath:

In computer processors, the datapath often consists of the following functional blocks, or some variation thereof:

- The instruction register stores the current instruction to be executed.
- The program counter (PC) stores the address of the next instruction to be fetched.
- The memory address register (MAR) is a register that either stores the memory address from which data will be fetched to the CPU or the address to which data will be sent and stored.
- The memory data register (MDR) is a register of a computer's control unit that contains the data to be stored in the computer storage (e. g. RAM), or the data after a fetch from the computer storage.

There are also two registers inherent in the processor that facilitate the communication of the processor with the memory, or basically help in the memory operations of the register.

INSTRUCTION CYCLE:

The basic function performed by computer is execution of program which consist of a set of instruction stored in memory. The processing required for a single instruction execution is called instruction cycle. Normally, instruction cycle include following sub-cycles:

a. **Fetch Cycle:** Read next instruction from memory into CPU.
b. **Execution Cycle:** Decode the Opcode and perform indicated operation.
c. **Interrupt Cycle:** If interrupts are enabled or an interrupt is executed, set the current process state and service for interrupt.

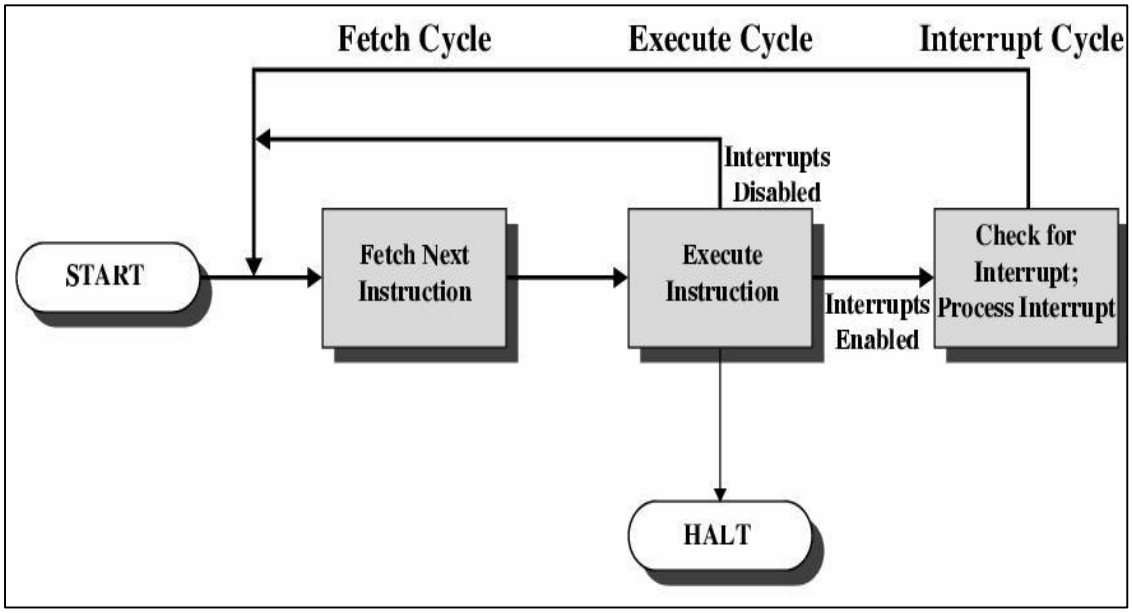


Fig: Instruction Cycle

The execution of an instruction may involve one or more operand in memory, each of which requires a memory access if indirect addressing is used then additional memory access is also required so, fetching of indirect address must be considered as extra cycle indirect execution.

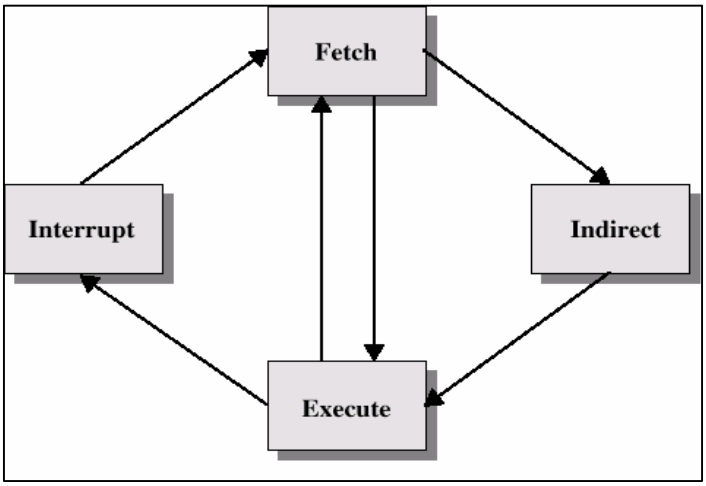


Fig: Instruction Cycle

INSTRUCTION CYCLE STATE DIAGRAM (DETAIL):

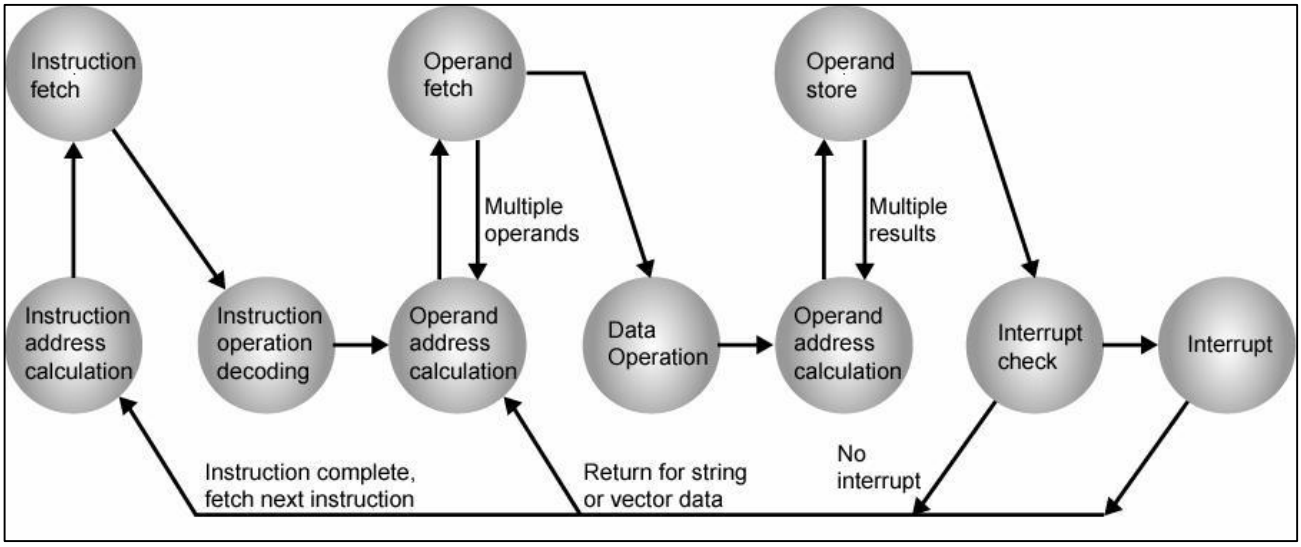


Fig: Instruction Cycle State Diagram

The figure above shown is in the form of state diagram. For any given instruction cycle some states may be NULL and others may be visited more than once. The states can be described as follows:

1. **Instruction Address Calculation:** Determine the address of the next instruction to be executed. Usually this involves adding a fixed number to the address of previous instruction.
2. **Instruction Fetch:** Read instruction from its memory location into the processor.
3. **Instruction Operand Decoding:** Analyze instruction to determine the type of operation to be performed and operations to be used.
4. **Operand Address Calculation:** If the operation involved reference to an operand in memory then determine the address of operand.
5. **Operand Fetch:** Fetch the operand from memory or read it in the form of input/output.
6. **Data Operation:** Perform the operation indicated in the instruction.
7. **Operand Store:** Write the result into memory or out to input/output.
8. **Interrupt Check:** If any type of error occurs in the program then service those interrupts.

ARITHMETIC AND LOGICAL UNIT:

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU).

Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A register is a small amount of storage available as part of a CPU. The control unit tells the ALU what operation to perform on that data, and the ALU stores the result in an output register. The control unit moves the data between these registers, the ALU, and memory.

An ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logic operations are comparisons of values such as NOT, AND, and OR.

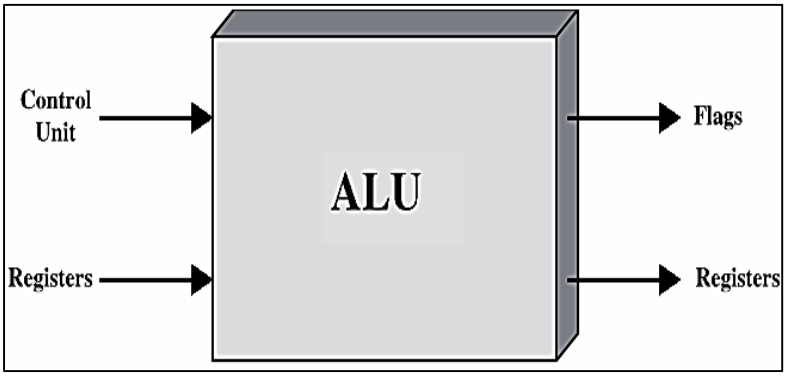


Fig: Block Diagram of ALU

DESIGN PRINCIPLES FOR MODERN SYSTEM:

There is a set of design principles, sometimes called the RISC design principles that architects of general-purpose CPUs do their best to follow:

1. **All Instructions Are Directly Executed by Hardware:**
   - Eliminates a level of interpretation

2. **Maximize the Rate at Which Instructions are Issued:**
   - MIPS = millions of instructions per second
   - MIPS speed related to the number of instructions issued per second
   - Parallelism can play a role

3. **Instructions Should be Easy to Decode:**
   - A critical limit on the rate of issue of instructions
   - Make instructions regular, fixed length, with a small number of fields.
   - The fewer different formats for instructions the better.

4. **Only Loads and Stores Should Reference Memory:**
   - Operands for most instructions should come from- and return to- registers.
   - Access to memory can take a long time
   - Thus, only LOAD and STORE instructions should reference memory.

5. **Provide Plenty of Registers:**
   - Accessing memory is relatively slow, many registers (at least 32) need to be provided, so that once a word is fetched, it can be kept in a register until it is no longer needed.

Chapter – 4

Computer Arithmetic

Data Representation:

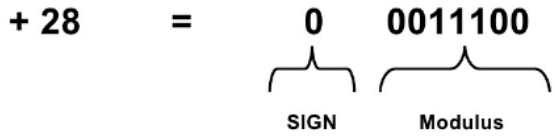
The data representation may be decimal, binary, octal, qunery, hexadecimal, etc. The data representation should have:

1. Provision For Decimal Position:

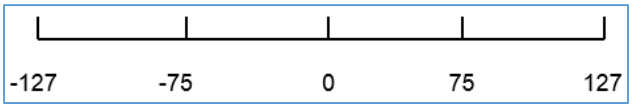
- **Fix Point Representation:** The decimal position is either at beginning or at the end of number. For example: 3.2 = 0.32*10
- **Floating Point Representation:** Here, second register is used to determine the position of decimal.

2. Provision For Sign Representation:

- **Signed Magnitude Representation:**
  This method uses the far left-hand bit which is often referred to as the sign bit. The general rule is that a positive number starts with 0 and a negative number starts with a 1.



- **One's Complement Representation:**
  It is the opposite of something. Because computer do not prefer to subtract, this method finds the complement of a positive number and then addition can take place.



Example

Find the one's complement of +100

1. Convert +100 to binary.
2. Swap all the bits.
3. Check answer by adding 127 to result.

Example

$+100 = 01100100_2$

$01100100_2 = 10011011_2$ (bit interchange)

$10011011_2 = -27_{10}$

$127_{10} + -27_{10} = 100_{10}$

- **Two's Complement Representation:**
  The only difference between the two processes is that the left most bit is -128_{10} rather than -127_{10}. The process for two's complement is exactly the same as one's complement, however we required to add 1 to the results.

  Example:
  Find the two's complement of $+15_{10}$
  1. Convert $+15_{10}$ to binary.
  2. Swap all the bits.
  3. Add 1 to the result.

  Example

  $+15 = 00001111_2$
  $00001111_2 = 11110000_2$ (Interchange)
  $11110000_2 + 1_2 = 11110001_2$
  $-128 + 64 + 32 + 16 + 1 = -15_{10}$

INTEGER ARITHMETIC:

1. NEGATION:

In sign-magnitude (1's Complement) representation, the rule for forming the negation of an integer is simple, just invert the sign bit. In 2's Complement notation, the negation of an integer can be formed with the following rules:

a. Take the Boolean complement of each bit of the integer (including the sign bit). That is, set each 1 to 0 and each 0 to 1.
b. Treating the result as an unsigned binary integer, add 1.

Among these two representation 2's Complement representation is the most widely used method.

2. ADDITION/SUBTRACTION:

Algorithm:

Step 1: Convert all the given numbers in binary form.
Step 2: If any number is negative then find its 2's Complement.
Step 3: Perform addition, if carry occurs then discard the carry.
Step 4: If sign bit of result is 1, then the result is negative. To find its magnitude re-complement the number.

Overflow:

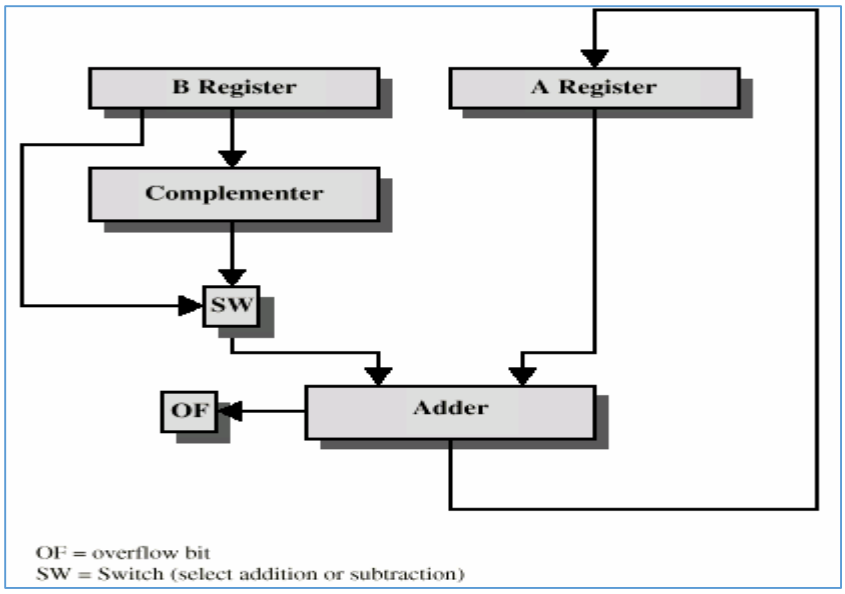
'N' bit operation produces more than N bits. This condition is called overflow condition.

Consider:

A ⟶ Register A

+ B ⟶ Register B

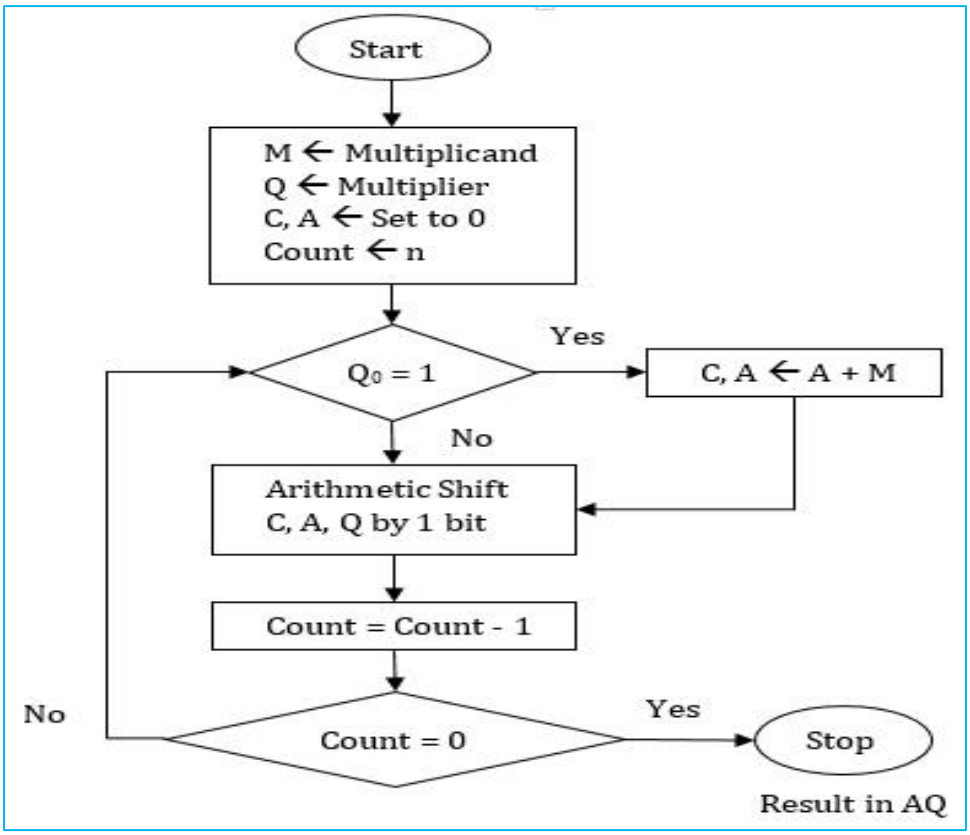
C



The central element is binary adder in which two numbers are presented for addition. The addition produces a sum and overflow condition. Two numbers are presented form two registers A and B. and finally the result is stored in accumulator. The overflow flag indicates, the overflow of data.

In above shown diagram if the number is negative then it is passed to adder after 2's Complement.

3. Unsigned Binary Multiplication Algorithm:



Perform unsigned multiplication of $(13)_{10} * (11)_{10}$

Solution:

$(13)_{10} = (1101)_2$

$(11)_{10} = (1011)_2$

C	A	Q	M	Count	Comment
0	0000	1101	1011	4	Initialize
0	1011	1101	1011	4	A ← A+M
0	0101	1110	1011	3	Right Shift C, A, Q
0	0010	1111	1011	2	Right Shift C, A, Q
0	1101	1111	1011	2	A ← A+M
0	0110	1111	1011	1	Right Shift C, A, Q
1	0001	1111	1011	1	A ← A+M
0	1000	1111	1011	0	Right Shift C, A, Q

Perform unsigned multiplication of $(8)_{10} * (3)_{10}$

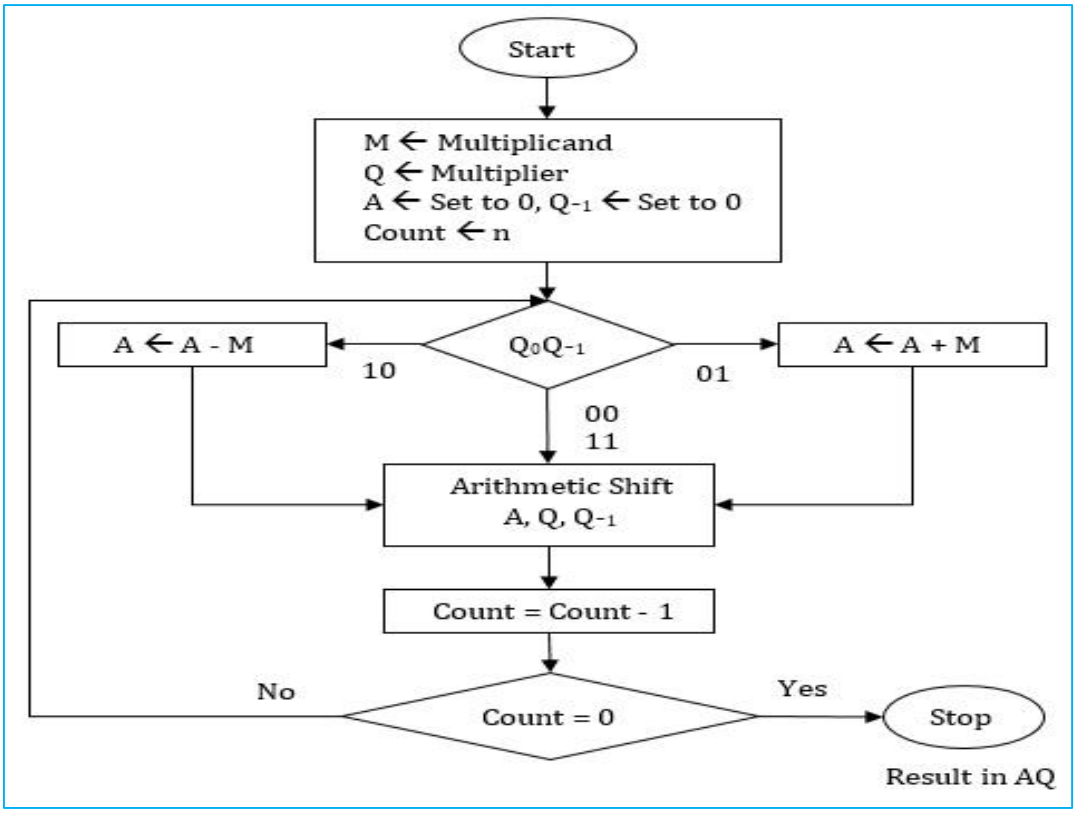
Solution:

$(8)_{10} = (1000)_2$

$(3)_{10} = (0011)_2$

C	A	Q	M	Count	Comment
0	0000	1000	0011	4	Initialize
0	0000	0100	0011	3	Right Shift C, A, Q
0	0000	0010	0011	2	Right Shift C, A, Q
0	0000	0001	0011	1	Right Shift C, A, Q
0	0011	0001	0011	1	A ← A+M
0	0001	1000	0011	0	Right Shift C, A, Q

4. Signed Binary Multiplication (Booth's) Algorithms:



Perform multiplication of $(7)_{10}$ * $(-3)_{10}$

Solution:

$(7)_{10} = (0111)_2$

$(-3)_{10} = (1101)_2$

A	Q	Q_{-1}	M	Count	Comment
0000	0111	0	1101	4	Initialize
0011	0111	0	1101	4	A ← A-M
0001	1011	1	1101	3	Right Shift A, Q, Q_{-1}
0000	1101	1	1101	2	Right Shift A, Q, Q_{-1}
0000	0110	1	1101	1	Right Shift A, Q, Q_{-1}
1101	0110	1	1101	1	A ← A+M
1110	1011	0	1101	0	Right Shift A, Q, Q_{-1}

Perform signed multiplication of $(6)_{10}$ * $(-4)_{10}$

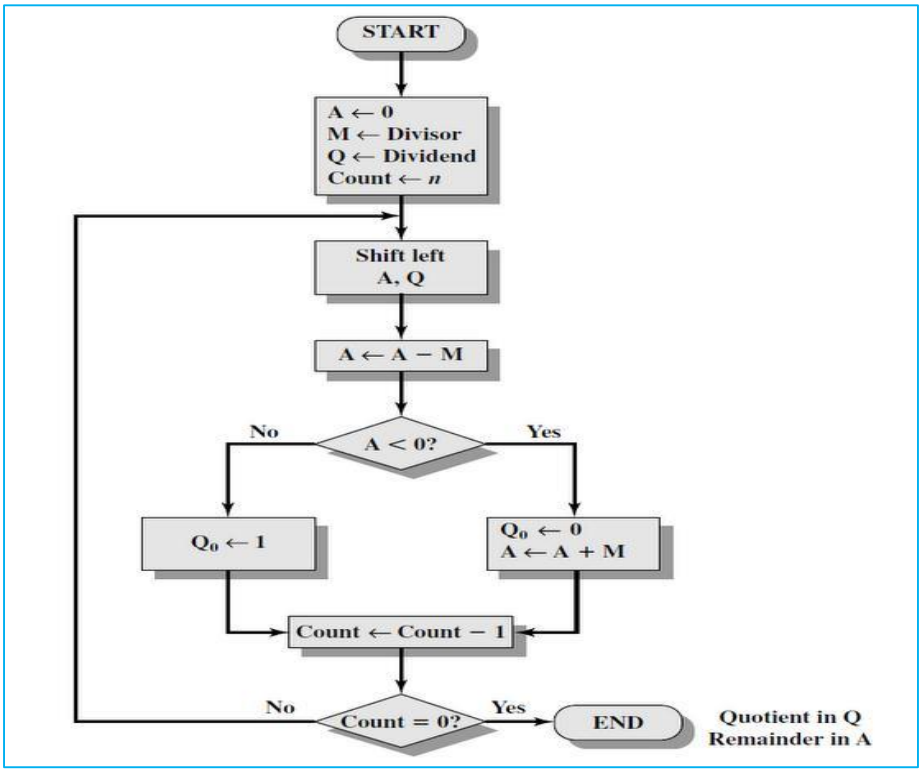
Solution:

$(6)_{10} = (0110)_2$

$(-4)_{10} = (1100)_2$

A	Q	Q_{-1}	M	Count	Comment
0000	0110	0	1100	4	Initialize
0000	0011	0	1100	3	Right Shift A, Q, Q_{-1}
0100	0011	0	1100	3	A ← A-M
0010	0001	1	1100	2	Right Shift A, Q, Q_{-1}
0001	0000	1	1100	1	Right Shift A, Q, Q_{-1}
1101	0000	1	1100	1	A ← A+M
1110	1000	0	1100	0	Right Shift A, Q, Q_{-1}

5. Unsigned Binary Division Algorithm:



Perform unsigned division of $(7)_{10}$ / $(3)_{10}$

Solution:

$(7)_{10} = (0111)_2$

$(3)_{10} = (0011)_2$

A	Q	M	Count	Comment
0000	0111	0011	4	Initialize
0000	1110	0011	4	Left Shift A, Q
1101	1110	0011	4	A ← A-M
0000	1110	0011	3	Q_0 ← 0, A ← A+M

0001	1100	0011	3	Left Shift A, Q
1110	1100	0011	3	A ← A-M
0001	1100	0011	2	Q_0 ← 0, A ← A+M
0011	1000	0011	2	Left Shift A, Q
0000	1000	0011	2	A ← A-M
0000	1001	0011	1	Q_0 ← 1
0001	0010	0011	1	Left Shift A, Q
1110	0010	0011	1	A ← A-M
0001	0010	0011	0	Q_0 ← 0, A ← A+M

Perform unsigned division of $(8)_{10}$ / $(4)_{10}$

Solution:

$(8)_{10} = (1000)_2$

$(4)_{10} = (0100)_2$

A	Q	M	Count	Comment
0000	1000	0100	4	Initialize
0001	0000	0100	4	Left Shift A, Q
1101	0000	0100	4	A ← A-M
0001	0000	0100	3	Q_0 ← 0, A ← A+M
0010	0000	0100	3	Left Shift A, Q
1110	0000	0100	3	A ← A-M
0010	0000	0100	2	Q_0 ← 0, A ← A+M
0100	0000	0100	2	Left Shift A, Q
0000	0000	0100	2	A ← A-M
0000	0001	0100	1	Q_0 ← 1
0000	0010	0100	1	Left Shift A, Q
1100	0010	0100	1	A ← A-M
0000	0010	0100	0	Q_0 ← 0, A ← A+M

REPRESENTATION OF FRACTIONS:

Fractional numbers are numbers between 0 and 1. They are called real numbers in computing terms. Real numbers contain both an integer and fractional part. *Example:* 23.714 OR 01101.1001. Computers use two main methods to represent real numbers:

a. Fixed Point Representation:

This method assumes the decimal point is in a fixed position. Relies on using a fixed number of bits for both the integer and fractional parts i.e. 5 and 3. *Example:* 10011101 = 10011.101

Integer					Fractional		
16	8	4	2	1	0.5	0.25	0.125
1	0	0	1	1	1	0	1

Therefore,

16 + 2 + 1 = 19 (Integer)
0.5 + 0.125 = 0.625 (Fraction)
So 10011.101 = 19.625

Problems are:

We fix the numbers that we can represent i.e. we are limited to amount of numbers that we can actually represent.

B. Floating Point Representation

This is the preferred method because we can represent large numbers. This uses exponential notation which highlights two specific parts of a decimal number:

- Mantissa: Fractional part.
- Exponent: Is the power by 10 which the mantissa is multiplied.

The aim of floating point representation is to show how many numbers before or after the decimal point. The general representation of large floating point representation is, $\pm S*B^{\pm E}$

Where,

- S is significant
- B is base
- E is exponent

A computer will represent a binary number into THREE parts:

- Sign Bit
- Mantissa
- Exponent

Example

$-241.65 = -0.24165 \times 10^3$
$0.0028 = 0.28 \times 10^{-2}$
$110.11 = 0.11011 \times 2^3$

Floating Point Arithmetic:

The overflow and underflow condition may occur in significant and exponent.

Addition and Subtraction:

Algorithm:

1. Check for zero. If anyone operand is zero then result will be another non-zero operand.

2. Align the significant. The decimal position can change so that the exponent value of both operand will be equal.
3. Add or subtract the significant.
4. Normalize the result.

The floating point numbers and their arithmetic calculation can be illustrated as:

Floating Point Numbers	Arithmetic Operations
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$\left.\begin{aligned} X+Y &= \left(X_S \times B^{X_E - Y_E} + Y_S\right) \times B^{Y_E} \\ X-Y &= \left(X_S \times B^{X_E - Y_E} - Y_S\right) \times B^{Y_E} \end{aligned}\right\} X_E \le Y_E$ $X \times Y = \left(X_S \times Y_S\right) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_S}{Y_S}\right) \times B^{X_E - Y_E}$

Examples:

$X = 0.3 \times 10^2 = 30$
$Y = 0.2 \times 10^3 = 200$

$X + Y = (0.3 \times 10^{2-3} + 0.2) \times 10^3 = 0.23 \times 10^3 = 230$
$X - Y = (0.3 \times 10^{2-3} - 0.2) \times 10^3 = (-0.17) \times 10^3 = -170$
$X \times Y = (0.3 \times 0.2) \times 10^{2+3} = 0.06 \times 10^5 = 6000$
$X \div Y = (0.3 \div 0.2) \times 10^{2-3} = 1.5 \times 10^{-1} = 0.15$

BCD ARITHMETIC UNIT:

Binary coded decimal (BCD) is a system of writing numerals that assigns a four-digit binary code to each digit 0 through 9 in a decimal (base-10) numeral. The four-bit BCD code for any particular single base-10 digit is its representation in binary notation, as follows:

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001

Numbers larger than 9, having two or more digits in the decimal system, are expressed digit by digit. For example, the BCD rendition of the base-10 number 1895 is 0001 1000 1001 0101. The binary equivalents of 1, 8, 9, and 5, always in a four-digit format, go from left to right.

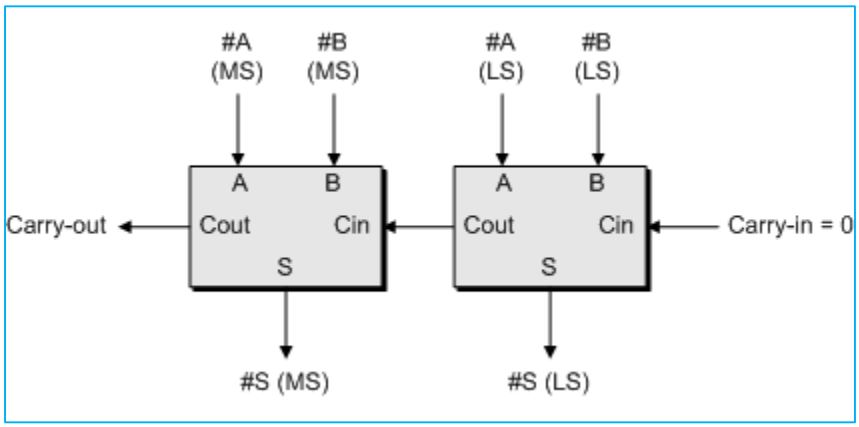
The BCD representation of a number is not the same, in general, as its simple binary representation. In binary form, for example, the decimal quantity 1895 appears as 11101100111

Other bit patterns are sometimes used in BCD format to represent special characters relevant to a particular system, such as sign (positive or negative), error condition, or overflow condition.

The BCD system offers relative ease of conversion between machine-readable and human-readable numerals. As compared to the simple binary system, however, BCD increases the circuit complexity. The BCD system is not as widely used today as it was a few decades ago, although some systems still employ BCD in financial applications.

BCD Adder:

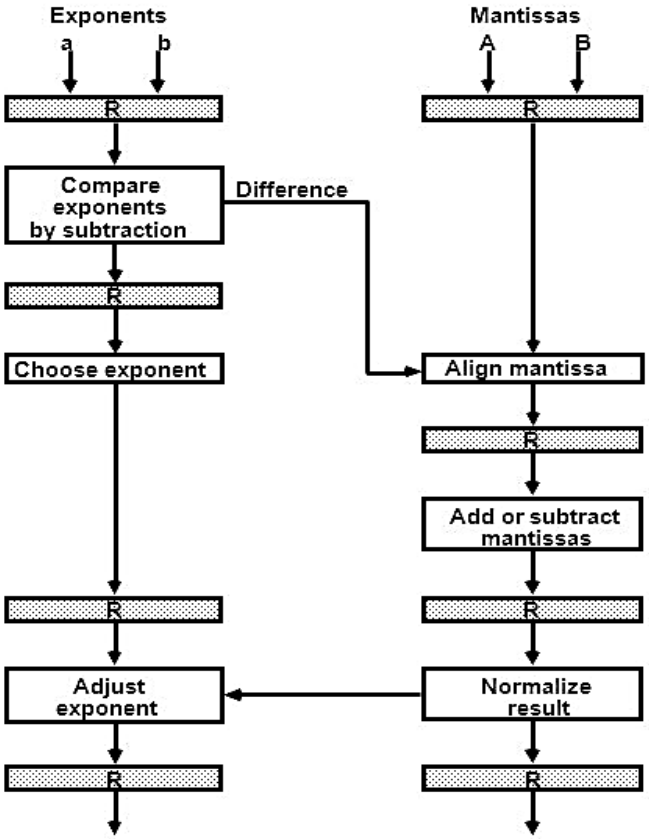
BCD adder is a circuit that adds two BCD digits in parallel and produces a sum digit also in BCD. A decimal parallel adder that adds n decimal digits needs n BCD adder stages with the output carry from one stage connected to the input carry of the next higher order stage.



ARITHMETIC PIPELINING:

Arithmetic pipelining is implemented to perform complex arithmetic calculations. Let us consider two variables whose values are represented in exponential form.

Let A = 2.343×10^{12} and B = 212.36×10^{10} then arithmetic addition and subtraction can be performed as follows. The operation is performed parallel.


Exponents
a
b
Mantissas
A
B
R
Compare exponents by subtraction
Difference
Choose exponent
Align mantissa
Add or subtract mantissas
Adjust exponent
Normalize result

Chapter – 5

Control Unit

Introduction:

The control unit (CU) is a component of a computer's central processing unit (CPU) that directs the operation of the processor. It tells the computer's memory, arithmetic/logic unit and input and output devices on how to respond to a program's instructions. It controls on primitive operations (micro-operations). The micro-operations for different sub-cycles are:

a. Fetch Cycle Micro-operation:

T1: MAR ← (PC)
T2: MBR ← Memory
PC +I; Instruction Length
T3: IR ← (MBR)

b. Indirect Cycle Micro-operation:

T1: MAR ← (IR Address)
T2: MBR ← Memory
T3: IR (Address) ← MBR (Address)

c. Interrupt Cycle Micro-operation:

T1: MBR ← (PC)
T2: MAR ← Save Address
PC ← Routine Address
T3: Memory ← (MBR)

d. Execute Cycle Micro-operation:

Add R1, X
R1 ← R1 + X
T1: MAR ← (IR (Address))
T2: MBR ← Memory
T3: R_1 ← (R_1) + (MBR)

Control of Processor/CPU:

❖ Responsibilities:

The basic responsibilities of control unit is to control various operations of CPU such as:

1. Data Exchange Of CPU With Memory Or Input/output Module:
2. Internal Operations Of CPU Such As:
   a. Moving data between Registers.
   b. Making ALU to perform particular operation on data.
   c. Regulating internal operations.

❖ Functional Requirement:

In order to define the functions of a control unit that it must perform, we have to know what resources and means the control unit need to perform those operations. A control unit must have following three requirements:

1. **Defining Basic Elements of Processor:**

The basic functional units of processor are ALU, Registers, Internal Data Path, External Data Path and Control Unit itself.

2. **Describe The Micro-Operations That The Processor Performs:**
   a. Transfer data from one register to another.
   b. Transfer data from external interface to register.
   c. Performing arithmetic logical and shift micro-operations.

3. **Determine The Functions That The CPU Must Perform To Cause Micro-Operation To Be Performed:**
   a. **Execution:** The control unit cause each micro-operation to be performed.
   b. **Sequencing:** It enables the CPU to execute proper sequence of micro-operations.

Structure of Control Unit with Control Signals:

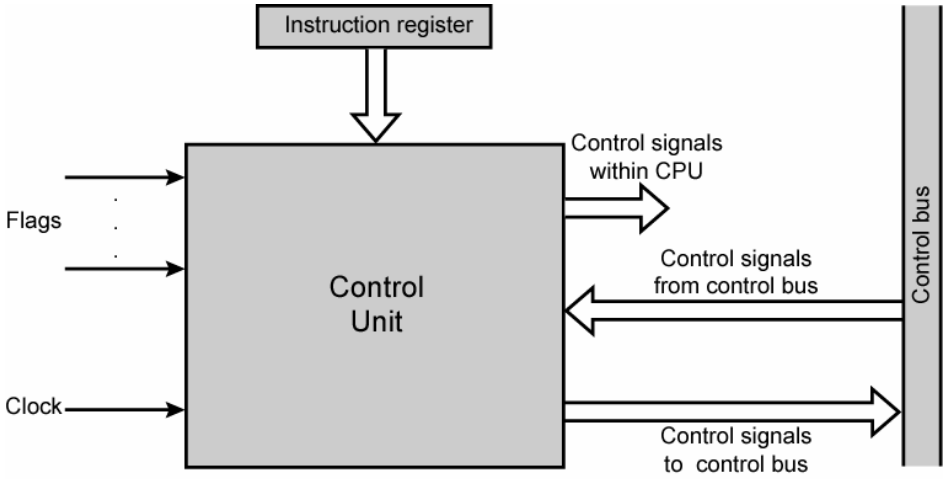


Fig: Block Diagram of Control Unit

For the control unit to perform its function it must have input that allows it to determine the state of system. These are the external specification of control unit. Internally control unit must have logics that is required to perform its sequencing and execution function. The input control unit are:

1. **Clock:**

This is how a control unit keeps time. The control unit cause one micro-operation or a set of simultaneous micro-operations to be performed for each clock pulse or system clock cycle time.

2. Flags:

These are needed by control unit to determine the status of processor and outcome of previous ALU operation.

3. Instruction Register:

The Opcode of current instruction is used to determine, which micro-operation is to be performed.

4. Control Signals from System Bus:

The control bus portion of system bus provides signal to control unit such as interrupt signal, acknowledgement signal, etc.

5. Control Signals within CPU:

These control signals cause two types of micro-operation such as data transfer from one register to another and performing various ALU operations using input/output and register.

6. Control Signals to System Bus:

The basic purpose of these control signals is to bring or transfer data from CPU register to memory or input/output module.

Types of Design Issues in Control Unit:

1. Hardwired Control Unit:

In a hardwired organization control logic is implemented with logic gates, flip-flop, decoder and other digital circuits with control bus signal. On the basis of these input the output signals are generated. Therefore, output control signals are the functions of input signals.

Advantages:

- It can be optimized to produce fast mode of operations.

Disadvantages:

- It is inflexible in design because of which it is more difficult to modify.
- It is quite difficult to test and implement because there are hundreds of control lines in the computer.

Two Main Issues in Hardwired Implementation:

a. Control Unit Input:

The main key inputs for control unit are instruction register, clock signals or time generator, flags and decoded instructions.

For flags and control bus signals, each bit has a typical meaning but in other two keys, instruction register and time generator extra circuit is required i.e. decoder.

The function of decoder is to produce more output from less number of input bits.

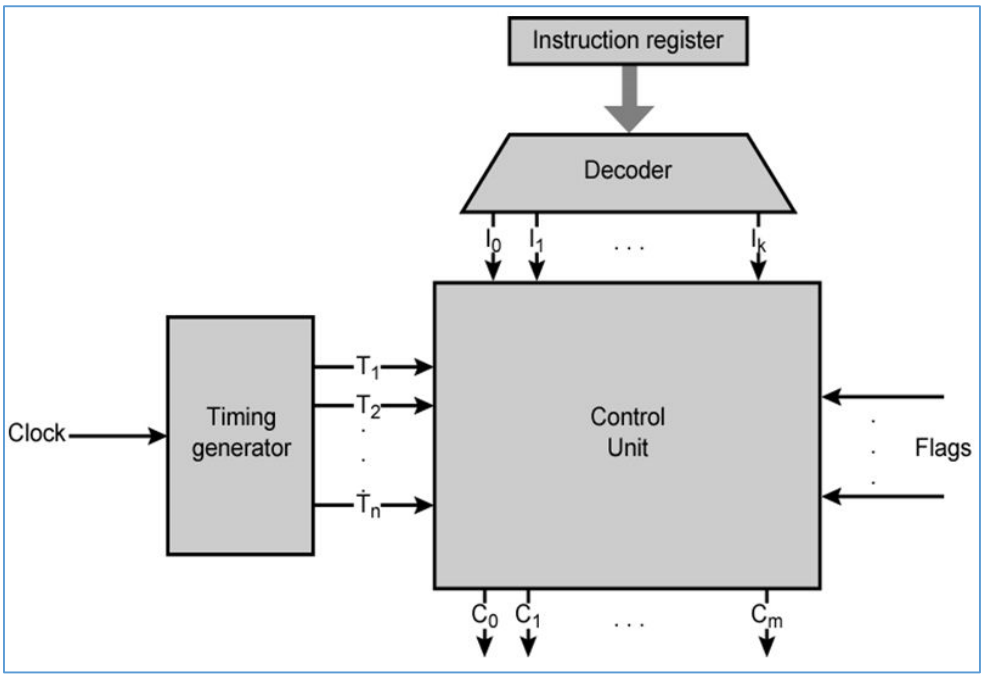


Fig: Control Unit with Decoded Input

b. <u>Control Unit Logic:</u>

Here each expression can be expressed (written) in the form of Boolean expression for each operation. The output control signals are as the function of input signals. Let us consider two control signals P and Q, which may be expressed as:

PQ = 00; Fetch Cycle
PQ = 01; Indirect Cycle
PQ = 10; Execute Cycle
PQ = 11; Interrupt Cycle

2. Micro-programmed Control Unit:

An alternative way to implement control unit is micro-program control unit. In which the logic of control unit is specified by micro-program. A micro-program consist of sequence of micro-instructions in a micro-programming language.

On executing these micro-instructions, they are responsible for execution of one or more micro-operation and sequencing through micro-operation.

A micro-program is also known as firmware or mid-way between hardware and software. In comparison to hardware, it is easier to design where as in comparison to software it is difficult to write. A micro-instruction is made responsible for generating control signal for desired control lines to implement a desired micro-operation.

A set of control signals with each bit representing a single control line is called control word. The micro-programs are stored in Read Only Memory (ROM) known as control memory.

Advantages:

- It is easy to design. So it is both cheaper and less error prone to implement.

- Since, micro-program can be changed relatively easily so, it is very flexible in comparison to hardwired control unit.

Disadvantages:

- It is slower than hardwired control unit because micro-instructions are to be fetched from control memory which is time consuming.

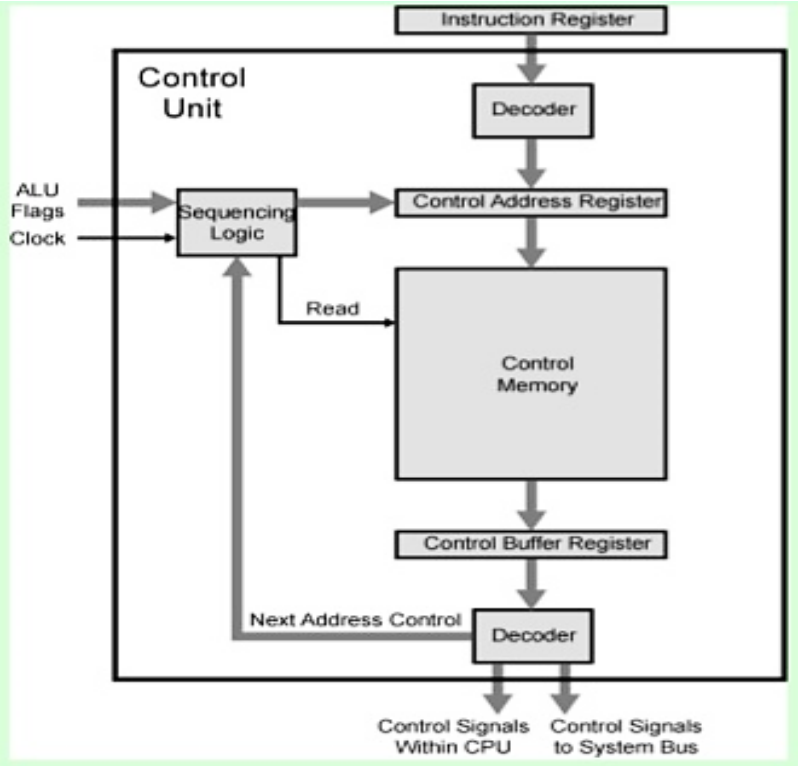


Fig: Micro programmed Control Unit

Micro-instruction Format:

The format of micro-instruction must include following fields:

- Micro-instruction address
- Jump condition, unconditional, zero, overflow, etc.
- System bus control signals
- Internal CPU control signals

Types of Micro-instruction Format:

1. **Horizontal Micro-instruction Format:**

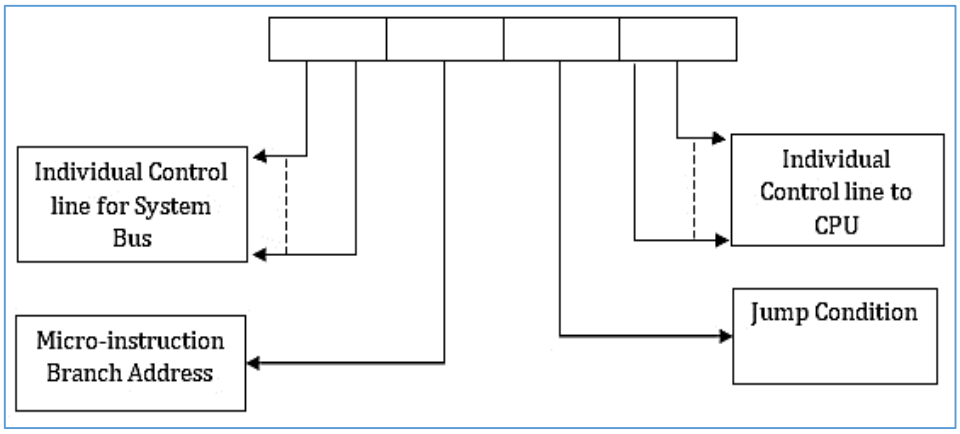


Fig: Horizontal Micro-instruction

In horizontal micro-instruction there is one bit for each internal processor control line and one bit for each system bus control line, therefore length of such microinstruction may be 100 of bits. Such micro-instructions may be executed as follows:

a. To execute micro-instruction turn ON all the control lines indicated by 1 and leave OFF all the control lines indicated by 0 bit. The resulting control signal will cause one or more micro-operations to be performed.
b. If the condition indicated by condition bit is FALSE execute next instruction in sequence.
c. If the condition indicated by condition bit is TRUE the next micro-instruction to be executed is indicated in address field.

2. **Vertical Micro-instruction Format:**

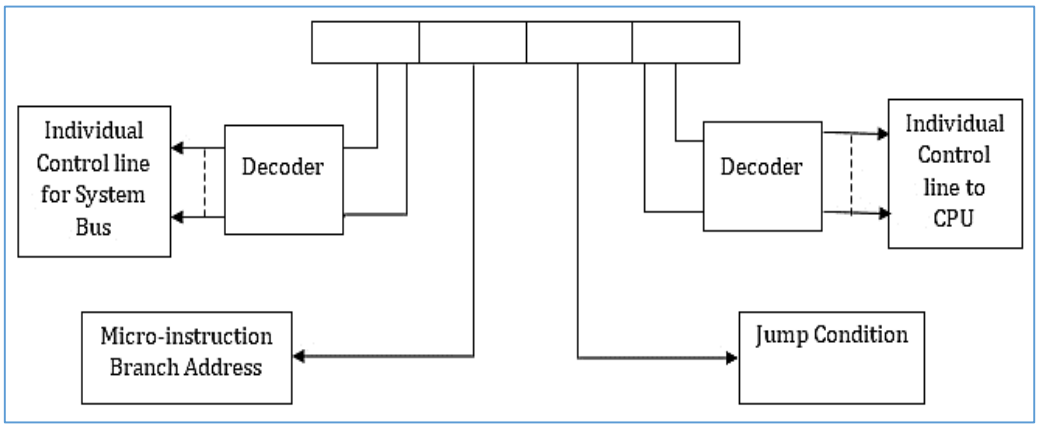


Fig: Vertical Micro-instruction

In vertical micro-instruction a code is used for each action to be performed and a decoder is used for translation of these codes into individual control signals. The advantages of vertical micro-instruction is that they are more complex and compact than horizontal microinstruction.

Difference between Horizontal and Vertical Micro-instruction Format:

Horizontal Micro-instruction Format	Vertical Micro-instruction Format
In a horizontal microinstruction every bit in the control field attaches to a control line.	In a vertical microinstruction, a code is used for each action to be performed and the decoder translates this code into individual control signals.
They do not have decoder.	They have decoder.
They need higher bits or wider bits.	They need fewer bits but higher number of decoder.
They are faster.	They are slower.
They are not complex and compact.	They are complex and compact.
They are easy to design.	They are hard to design.

Difference between Hardwired and Micro-programmed Control Unit:

Hardwired Control Unit	Micro-programmed Control Unit
It uses flags, decoder, logic gates and other digital circuits.	It uses sequence of micro-instruction in micro programming language.
As name implies it is a hardware control unit.	It is mid-way between Hardware and Software.
On the basis of input Signal output is generated.	It generates a set of control signal on the basis of control line.
Difficult to design, test and implement.	Easy to design, test and implement.
Inflexible to modify.	Flexible to modify.
Faster mode of operation.	Slower mode of operation.
Expensive and high error.	Cheaper and less error.
Used in RISC processor.	Used in CISC processor.

Microinstruction Sequencing:

Sequencing means getting next microinstruction from control memory. Two consequences are involved in design of microinstruction sequencing techniques and they are:

a. Size Of Microinstruction
b. Address Generation Time

The first issues try to minimize the size of control memory to reduce the cost of control memory component. Similarly, the second issues is used to develop or design a microinstruction that can be executed as fast as possible.

In executing a microinstruction the address of next microinstruction to be executed is one of the following categories:

a. Determine by instruction register
b. Next sequential address
c. Branch address

Microinstruction Sequencing Techniques:

Based on the current microinstruction condition (content of flag and content of instruction register). A control memory address must be generated for next microinstruction. In general three techniques based on the number of address field in microinstruction is utilized for sequencing.

1. **Two Address Field Microinstruction:**

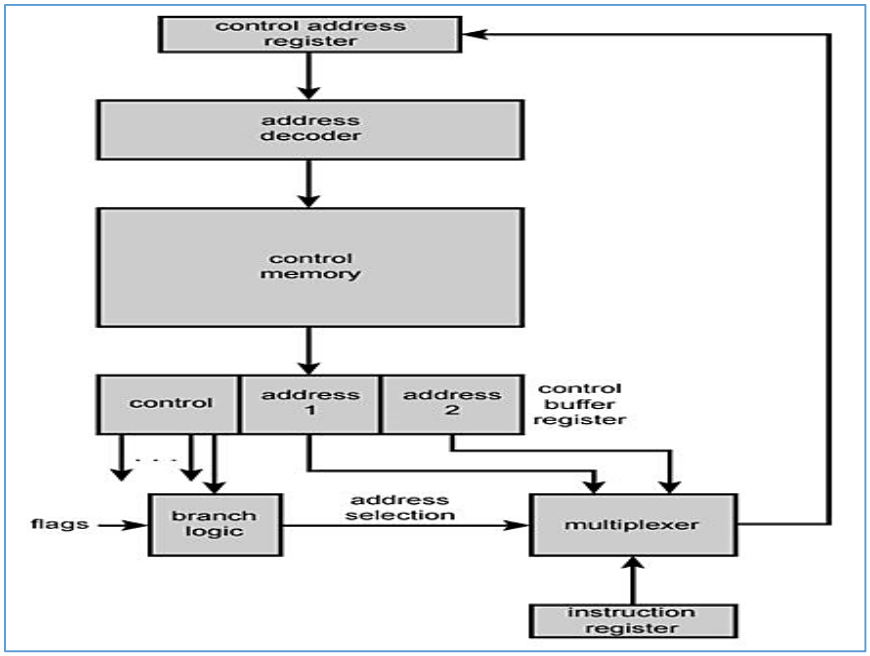


Fig: Branch Control Logic: Two Address Fields

The simplest approach is to provide two address field in each microinstruction. Above figure suggest how these information is to be used. A multiplexer is provided that serves as a destination for both address fields plus the instruction register. On the basis of address selection input, the multiplexor transmits either the Opcode or one of the two address to Control Address Register (CAR).

Address selection signals are provided by branch logic module whose input consist of control unit flags plus bits from control portion of microinstruction. Although two address approach is simple, it requires more bits in microinstruction than other approach.

2. Single Address Field Microinstruction:

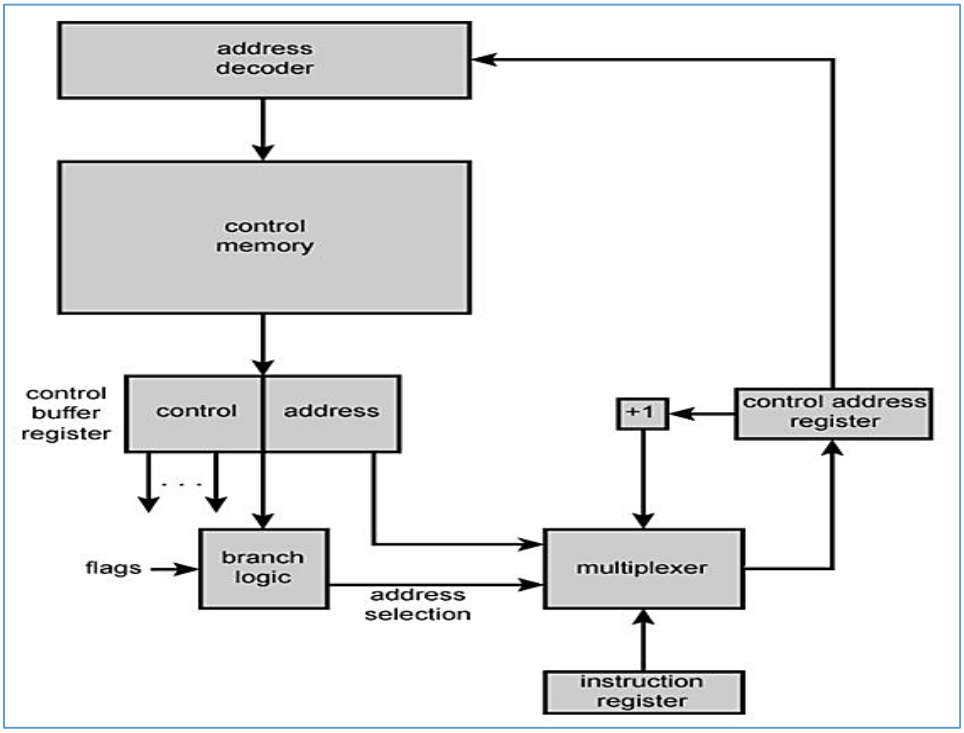


Fig: Branch Control Logic: Single Address Field

Single Address Field approach is common approach in which option for next address will be one of the following address:

a. Address Field
b. Instruction Register Code
c. Next Sequential Address

The address selection signal determine which option is selected. The approach reduces the number of address field to one.

3. Variable Format Microinstruction:

In variable format branch control logic one bit designates which format is being used. In one format, the remaining bits are used to active control signals. In the other format, some bits drive the branch logic module, and the remaining bits provide the address. With the first format, the next address is either the next sequential address or an address derived from the instruction register. With the second format, either a conditional or unconditional branch is being specified.

One disadvantage of this approach is that one entire cycle is consumed with each branch microinstruction.

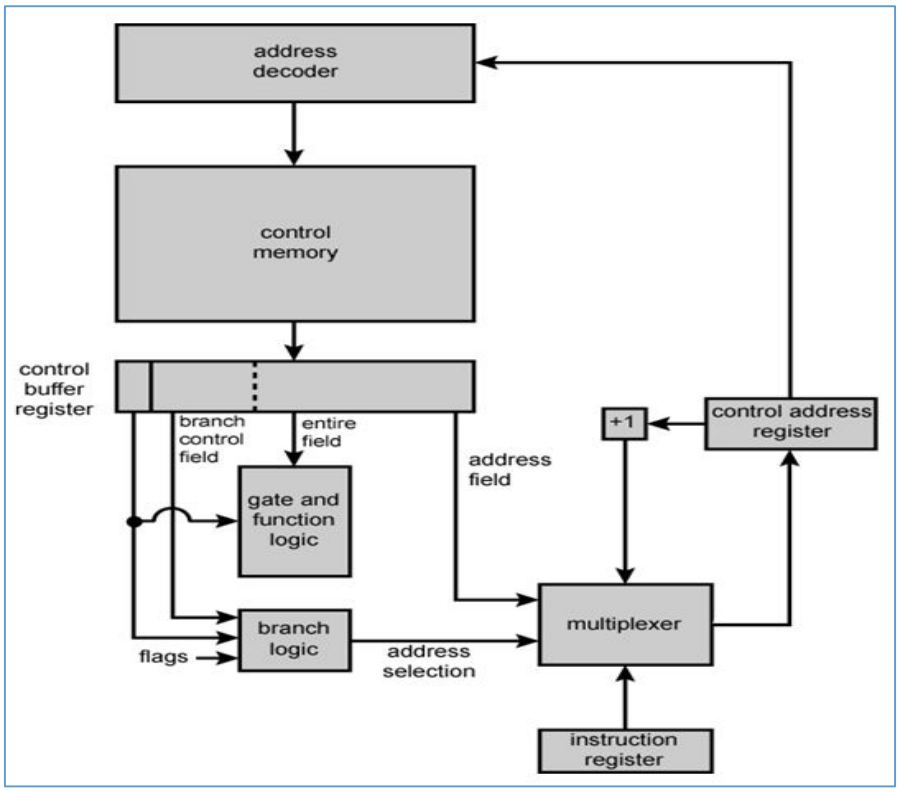


Fig: Branch Control Logic: Variable Format

MICRO-INSTRUCTION EXECUTION:

The microinstruction cycle is the basic event on a micro-programmed processor. Each cycle is made up the two parts: fetch and execute. This section deals with the execution of microinstruction. The effect of the execution of a microinstruction is to generate control signals for both the internal control to processor and the external control to processor.

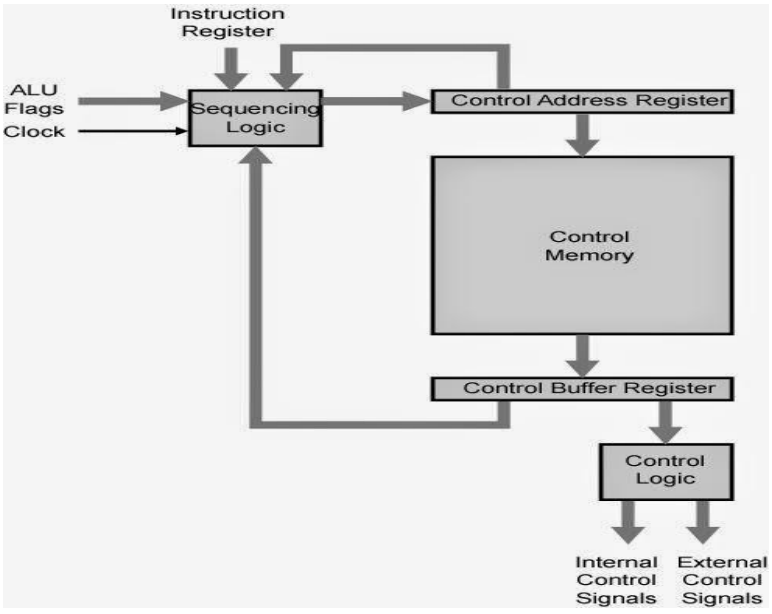


Fig: Control Unit Organization

Chapter – 6

Memory Organization

Memory Hierarchy:

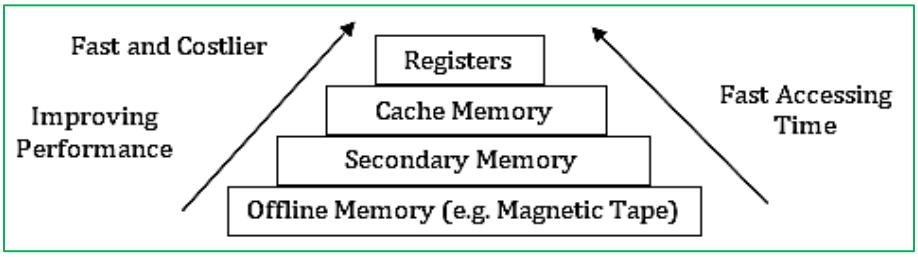


Fig: Memory Hierarchy

To achieve greatest performance the memory must be able to keep up with the processor. The designer would like to use memory technologies that provide larger capacity memory because capacity is needed for better performance of the system.

A typical hierarchy is shown in figure above. As we go down to the hierarchy the following features occurs:

a. Decreasing cost per bit.
b. Increasing access time.
c. Decreasing frequency of access of the memory by the processor.

Thus, smaller, more expensive, faster memory are supplemented by larger, cheaper and slower memory. The key to the success of this organization is decreasing the frequency of access.

Semiconductor Main Memory:

The basic element of semiconductor memory is the memory cell. Although a variety of electronic technologies are used, all semiconductor memory cells share certain properties:

a. They exhibit two stable (or semistable) states, which can be used to represent binary 1 and 0.
b. They are capable of being written into (at least once), to set the state.
c. They are capable of being read to sense the state.

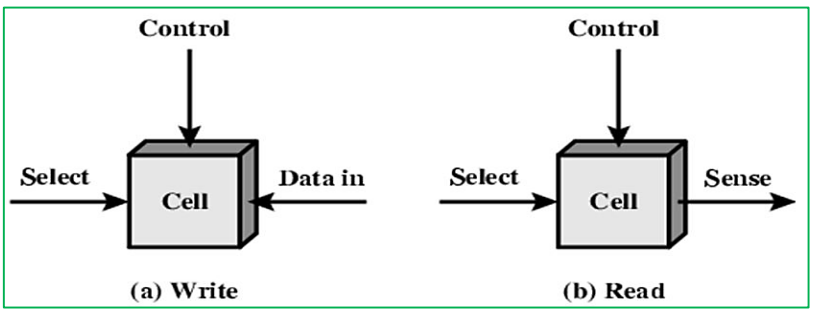


Fig: Memory Cell Operation

Figure above shows the operation of memory cell. Most commonly the cell has three functional terminals capable of carrying an electrical signal. The select terminal, selects a memory cell for

a read or write operation. The control terminal indicates read or write. For writing, the other terminal provides an electrical signal that sets the state of the cell to 0 or 1. For reading, that terminal is used for output of the cell state.

RAM (Random Access Memory):

RAM stands for "Random Access Memory" and is often called primary/main memory because it is made up of semiconductor chips. It is the working space used by the computer to hold the program that is currently running along with the necessary data and instructions. It is fast and expensive memory which allows the computer to access the data and instructions very quickly.

We can read from RAM as well as write into it. Hence is also called "Read-Write" memory. The main drawback of RAM is that it is volatile memory so the contents of RAM are lost when the computer is switched off.

It is made of millions of microscopic cells which are distinctly numbered so that each cell can be identified and located. Each cell can be electrically charged or not. The charged cell represents 1 and not charged cell represents 0 in binary format. RAM is also of two types:

1. DRAM:

DRAM stands for "Dynamic Random Access Memory". It is made up of capacitors which is capable of storing the electric charge. Due to the leakage of charges, the capacitors discharge gradually and the memory cells lose their contents. So, to recharge the capacitors to retain its memory contents it has to be refreshed periodically. DRAM is slower than SRAM but it is dense, consumes less electricity, smaller in size and less expensive.

Synchronous Dynamic Random Access Memory "SDRAM" is DRAM that has a synchronous interface which is widely used in present computers. Traditionally, DRAM has a synchronous interface, which means that it control inputs. SDRAM has a synchronous interface, meaning that it waits for a clock signal before synchronized with the computers system bus.

Example: DDR "Dual Data Rate", DDR2, DDR3, EDO DRAM, SDRAM, RIMM, etc.

2. SRAM:

SRAM stands for "Static Random Access Memory" and is made up of transistors. It is called static because it can remember or retain its memory contents without being refreshed or recharged as long as there is power. SRAM is faster than DRAM but more expensive, loser in density and bigger in size and consumes more electricity.

ROM (Read Only Memory):

ROM stands for "Read Only Memory" and it is called ROM because only read operation can be performed on it. The binary information stored in ROM is written permanently by the manufacturer and it cannot be altered. ROM is necessary to store such software which enables the computer to boot up because booting instructions does not need modification.

ROM is non-volatile memory because it can retain its contents even after the computer is turned off. It is also made semiconductor chips. The program stored permanently in ROM is called firmware. Hence, firmware is immediately available when a device is powered on to start up the PC or other electronic equipment like mobile, PDA and others. ROM is of three types:

1. PROM:

PROM stands for "Programmable Read Only Memory". Initially it is the blank chip which can be written or programmed only one time by using a special machine called ROM programmer or ROM burner. Once the PROM is written, it cannot be modified and becomes ROM.

2. EPROM:

EPROM stands for "Erasable Programmable Read Only Memory". It is a special chip which can be re-programmed to record different information. The data and information are erased by exposing it to intense ultra violet light for about 20 minutes. These types of memory are used in product development and experimental projects.

3. EEPROM:

EEPROM stands for "Electrically Erasable Programmable Read Only Memory". These types of chips can be erased and re-programmed repeatedly with special electrical pulses. It does not require a special device to write into it. EEPROM can be re-programmed without removing it from the computer. It also has limited life span i.e. the number of times it can be re-programmed is limited to tens or hundreds or thousands of times.

Differentiate between RAM and ROM:

RAM	ROM
RAM stands for Random Access Memory.	ROM stands for Read Only Memory.
RAM is volatile memory, if power fails data and information will be lost.	ROM is inherently non-volatile memory, if power fails data and information will not be lost.
RAM is used for currently running programs of computer system.	ROM is used to store firmware of computer system and system software for embedded system.
RAM is read/write memory.	ROM is read only memory.
The cost of RAM is higher than ROM.	The cost of ROM is lower is than RAM.
There are two types of RAM: SRAM and DRAM.	There are three types of ROM: PROM, EPROM and EEPROM

Differentiate between DRAM and SRAM:

DRAM	SRAM
DRAM stands for Dynamic Random Access Memory.	SRAM stands for Static Random Access Memory.
DRAM is made up of capacitors.	SRAM is made up of transistors.
DRAM is high density RAM. In one chip larger memory can be constructed.	SRAM is low density of RAM. In one chip small memory can be constructed.
Power consumption of DRAM is higher than SRAM.	Power consumption of SRAM is lesser than DRAM.
DRAM need to be periodically refreshed. (System automatically refreshed the RAM cells.)	SRAM does not need to be periodically refreshed.
The cost of DRAM is lower than SRAM.	The cost of SRAM is higher than DRAM.

The data access time is larger than SRAM, typically requires larger than 40 nanoseconds. Hence they are slow.	The data access time is smaller than DRAM, typically less than 30 nanoseconds. Hence they are fast.
DRAM is generally used for low cost high capacity memory for computers.	SRAM is generally used to create memory of critical section like cache memory.
Example: DDR, DDR2, DDR3 (Dual Data Rate), EDO DRAM, SDRAM, RIMM, etc.	Example: cache memory of microprocessor.

CHARACTERISTICS OF MEMORY SYSTEM:

1. **LOCATION:**
   Internal (Example: Registers, Main Memory, and Cache)
   External (Example: Magnetic Disk, Optical Disk, etc.)

2. **CAPACITY:**
   Number of words
   Number of bytes

3. **UNITS OF TRANSFER:**
   Word
   Block

4. **ACCESS METHOD:**
   Sequential, Direct
   Random, Associative

5. **PERFORMANCE:**
   Access Time
   Cycle Time
   Transfer Rate

6. **PHYSICAL CHARACTERISTICS:**
   Volatile or Non-volatile
   Erasable or Non-erasable

AUXILIARY MEMORY:

1. **MAGNETIC DISK:**

A disk is a circular platter constructed of metal or of plastic coated with magnetic materials. Data are recorded on and later retrieved from the disk through a conducting coil known as head. During a read or write operations the head is stationary while the platter rotates it. Writing is achieved by producing a magnetic field which records a magnetic pattern on magnetic surface.

The figure below shows the data layout of disk. The head is capable of reading or writing. The data is transferred to and from the disk in blocks. The sectors may be of fixed or variable length. The platter in a disk may be of single or multiple.

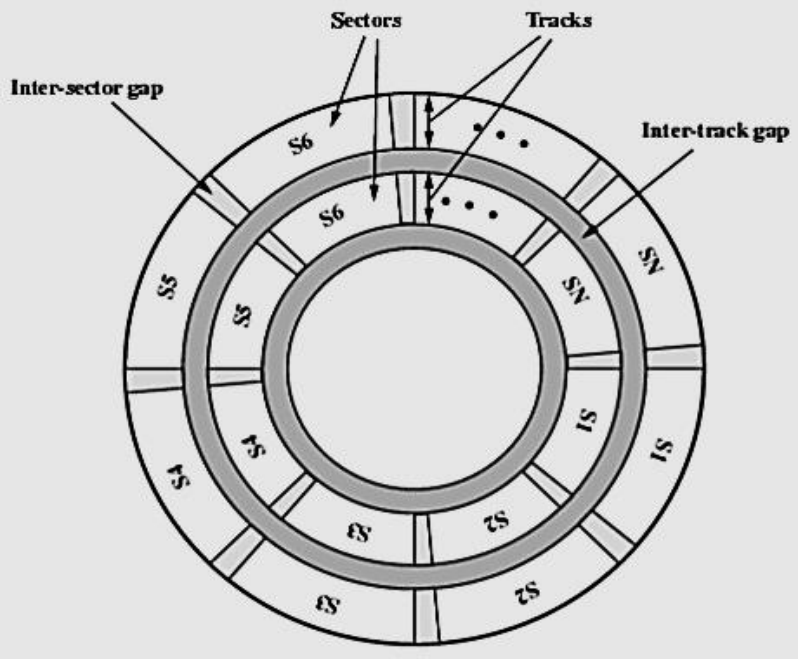


Fig: Disk Data Layout

2. Magnetic Tape:

Tape systems use same reading and recording techniques as disk systems. The way of recording the data is parallel. It stores one bytes of data at a time. Data on the tape are structured as a number of parallel tracks running lengthwise.

The data are read and written in contiguous track called physical records. Blocks on the tape are separated by gaps called interrecord gaps. It is a system for storing digital information on tape using digital recording. The device that performs writing or reading of a data is a tape drive.

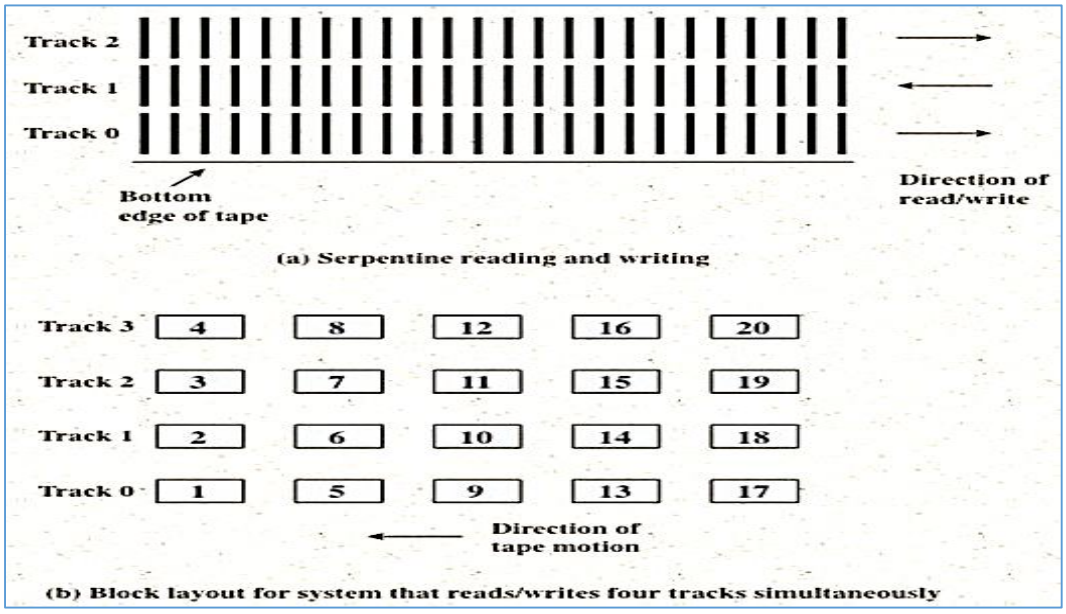


Fig: Magnetic Tape Features

3. Optical Disk:

An optical disc is an electronic data storage medium that can be written to and read using a low-powered laser beam. Originally developed in the late 1960s, the first optical disc, created by James T. Russell, stored data as micron-wide dots of light and dark. A laser read the dots, and the data was converted to an electrical signal, and finally to audio or visual output. However, the technology didn't appear in the marketplace until Philips and Sony came out with the compact disc (CD) in 1982. Since then, there has been a constant succession of optical disc formats, first in CD formats, followed by a number of DVD formats.

Optical disc offers a number of advantages over magnetic storage media. An optical disc holds much more data. The greater control and focus possible with laser beams (in comparison to tiny magnetic heads) means that more data can be written into a smaller space. Storage capacity increases with each new generation of optical media. Emerging standards, such as Blu-ray, offer up to 27 gigabytes (GB) on a single-sided 12-centimeter disc. In comparison, a diskette, for example, can hold 1.44 megabytes (MB). Optical discs are inexpensive to manufacture and data stored on them is relatively impervious to most environmental threats, such as power surges, or magnetic disturbances.

4. Flash Drives:

A small, portable flash memory card that plugs into a computers USB port and functions as a portable hard drive. USB flash drives are touted as being easy-to-use as they are small enough to be carried in a pocket and can plug into any computer with a USB drive. USB flash drives have less storage capacity than an external hard drive, but they are smaller and more durable because they do not contain any internal moving parts.

USB flash drives also are called thumb drives, jump drives, pen drives, key drives, tokens, or simply USB drives.

A flash drive consists of a small printed circuit board carrying the circuit elements and a USB connector, insulated electrically and protected inside a plastic, metal, or rubberized case which can be carried in a pocket or on a key chain. Most flash drives use a standard type-A USB connection allowing connection with a port on a personal computer, but drives for other interfaces also exist.

5. Review Of RAID (Redundant Array Of Independent Disks):

RAID is a set of physical disk drive viewed by operating system as a single logical drive. Data are distributed across the physical drive of an array in a scheme known as striping. Redundant disk capacity is used to store parity information which guarantee data recoverability in case of disk failure. There are different levels of RAID and they are:

a. RAID 0:

- Often called striping
- Break a file into blocks of data
- Simple to implement
- Provides no redundancy or error detection

b. RAID 1:

- Complete file is stored in a single disk

- A second disk contains an exact copy of file
- Provides complete redundancy of data

c. RAID 2:

- Strips data across disk similar to level – 0
- A parity disk is used to reconstruct corrupted or lost data.
- Uses Error Checking Code (ECC) to monitor correctness of information

d. RAID 3:

- It eliminates the problem of level – 2 that is disk need to detect which disk has an error.
- Modern disk can already determine if there is an error.

e. RAID 4:

- It consists of a block level striping with dedicated parity.
- It allows multiple small input/output to be done at once.

f. RAID 5:

- It consists of block level striping with distributed parity.
- Here, also parity information is distributed among the drives.

g. RAID 6:

- Consists of block level striping with double distributed parity.
- Double parity provides fault tolerance upto two failed drives.

ASSOCIATIVE MEMORY:

1. HARDWARE ORGANIZATION:

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 address. A memory unit accessed by the content is called Associative Memory or Content Accessible Memory. This type of memory is accessed simultaneously and parallel on the basis of data content rather than by specific address or location. When a word is written in associative memory no address is given.

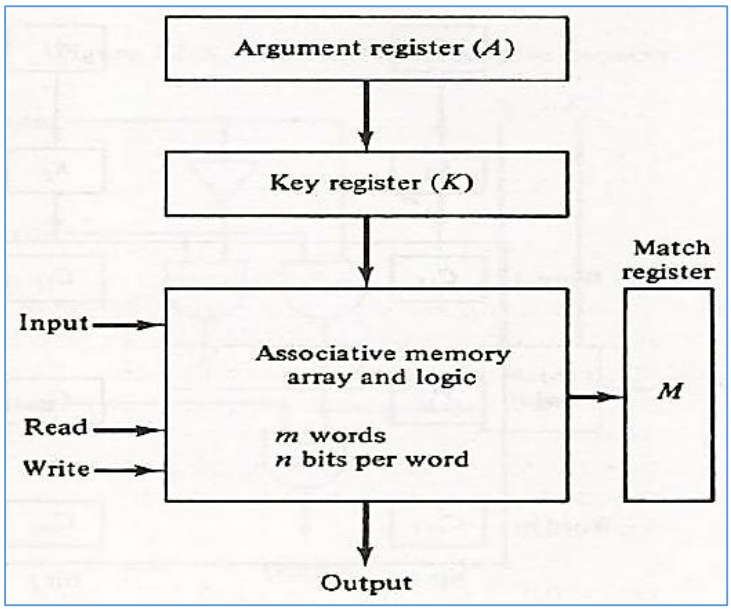


Fig: Block Diagram of Associative Memory

Associative Memory is organized in such a way:

a. **Argument Register (A):** It contains the word to be searched. It has 'n' bits (one for each bit of the word).
b. **Key Register (K):** This specifies which part of the argument word needs to be compared with words in memory. If all bits in register are 1, the entire word should be compared. Otherwise, only the bits having K-bits set to 1 will be compared.
c. **Associative Memory Array:** It contains the words which are to be compared with the argument word.
d. **Match Register (M):** It has 'm' bits, one bit corresponding to each word in the memory array. After the matching process, the bits corresponding to matching words in match register are set to 1.

2. Address Matching Logic:

Key register provide the mask for choosing the particular field in A register. The entire content of A register is compared if key register content all 1. Otherwise, only bit that have 1 in key register are compared. If the compared data is matched corresponding bits in the match register are set. Reading is accomplished by sequential access in memory for those words whose bit are set. Example:

A	101 111100	
K	111 000000	
Word 1	100 111100	no match
Word 2	101 000001	match

Let us include key register. If Kj = 0 then there is no need to compare Aj and Fij. Only when kj = 1, comparison is needed. This achieved by ORing each term with Kj. Mi = $(X_1 + K'_1)$ $(X_2 + K'_2)$ $(X_3 + K'_3)$…. (Xn + K'n)

3. Read/Write Operations:

Read Operations:

When a word is to be read from an associative memory, the contents of the word, or a part of the word is specified. If more than one word match with the content, all the matched words will have 1 in the corresponding bit position in match register. Matched words are then read in sequence by applying a read signal to each word line. In most application, the associative memory stores a table with no two identical items under a given key.

Write Operations:

If the entire memory is loaded with new information at once prior to search operation then writing can be done by addressing each location in sequence. Tag register contain as many bits as there are words in memory. It contains 1 for active word and 0 for inactive word. If the word is to be inserted, tag register is scanned until 0 is found and word is written at that position and bit is change to 1.

4. Types of Associative Memory:

There are two types of Associative Memory, which both are used in different conditions.

a. Auto-Associative:

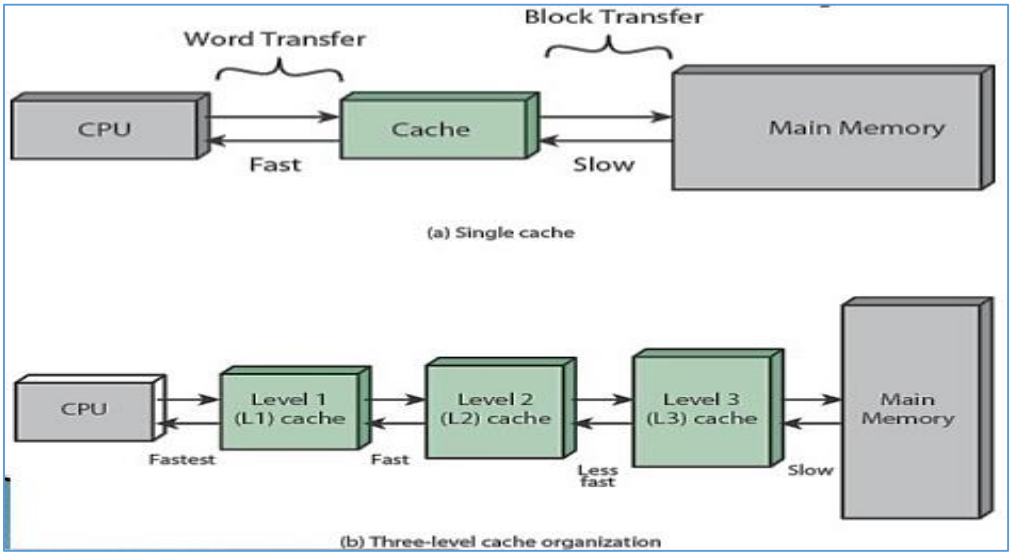
Auto-associative memory takes back (retrieves) a previously stored pattern that most closely resembles the current pattern.

b. Hetero-Associative:

Hetero-associative memory, the retrieved pattern is in general, different from the input pattern not only in content but possibly also in type and format. Neutral network are used to implement these associative memory models called NAM (Neutral Associative Memory).

Cache Memory:

1. Cache Initialization:



The cache contains a copy of the portion main memory. When the processor attempts to read a word of memory a check is made to determine if the word is in the cache. If so, the word is delivered to the processor if not block of main memory is read and transfer to the cache and is delivered to the processor.

Because of the phenomena of locality of reference, when block of data is fetched into a cache it is likely that there will be a future reference to the same memory location. As shown in figure above a block of data is transferred between cache and main memory where as word of data is transferred between CPU and Cache.

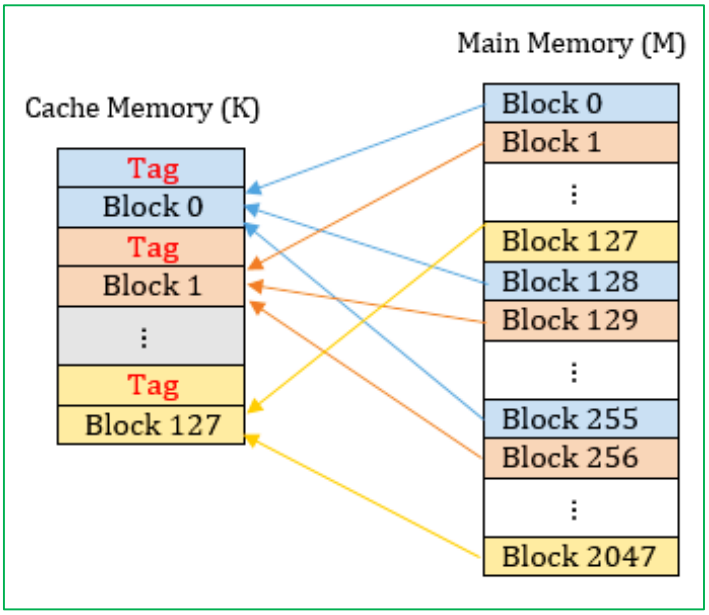
Different levels of caches can be created on the basis of uses. For example on the basis of speed level 1 cache is faster than level 2 and level 2 is faster than level 3.

The performance of cache memory is measured in terms of quantity called hit ratio. **When the CPU refers to memory and finds the word in cache it is said to produce hit. If the word is not found in cache it counts a miss. The ratio of number of hits divided by total CPU references to memory is called hit ratio.**

2. Mapping Cache Memory:

Transformation of data from main memory to cache memory is known as mapping process. There are 3 types of mapping process and they are:

a. Direct Mapping:

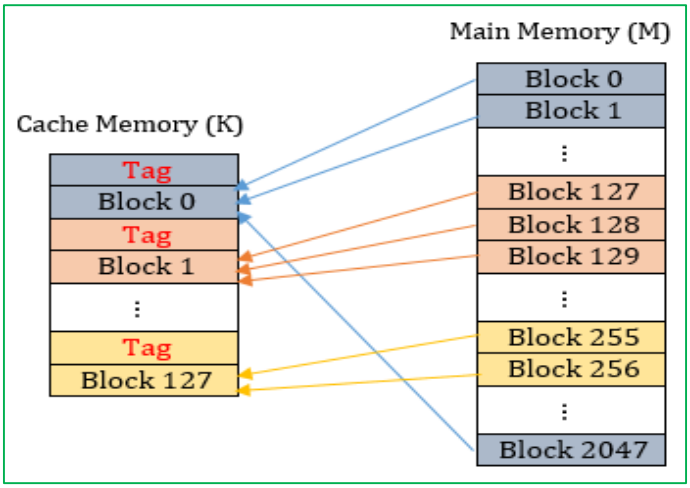


This is the simplest mapping technique in which bock 'M' of main memory is mapped into block 'K' of cache memory. Since, more than one main memory block is mapped into a given cache block position contention may arise for that position even if the cache is not full.

A main memory address can be divided into 3 fields i.e. tag, block and word. The tag bit is required to identify a main memory block when it is resident in cache. When a new block enters the cache, the cache block field determines the cache position. The main memory address can be divided into 3 fields are:

Main Memory Address		
5	7	4
Tag	Block	Word

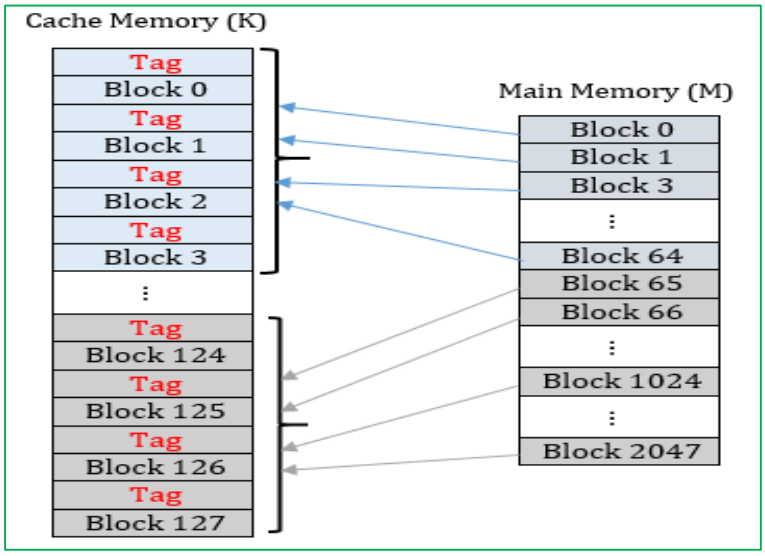
b. Associative Mapping:



This is much more flexible mapping technique, here any main memory block can be loaded to any cache block position. In this case, 12 tag bits are required to identify main memory block. The tag bits of an address received from CPU are compared with the tag bits of each cache blocks to see if the desire block is present in the cache. The main memory address can be divided into 2 fields are:

Main Memory Address	
12	4
Tag	Block

c. Block Set Associative Mapping:



In block set associative mapping technique, blocks of the cache are grouped into sets and the mapping allows a block of main memory to reside in any block of particular set. As shown in diagram above a cache with four block per set is used for mapping technique.

The six bit set field of the cache might contain the address block. As shown in diagram above two kilobyte of main memory is used to transfer its data to cache memory. The contention problem of the direct method is overcome by having few choices for block replacement.

Similarly, the hardware cost is reduced by reducing the size of associative search which are its advantages. The main memory address can be divided into 3 fields are:

Main Memory Address		
6	6	4
Tag	Block	Word

3. Write Policy:

When a system writes data to cache, it must at some point write that data to the backing store as well. The timing of this write is controlled by what is known as the **write policy**.

There are two basic writing approaches:

a. **Write-through:** write is done synchronously both to the cache and to the backing store.
b. **Write-back (also called write-behind):** initially, writing is done only to the cache. The write to the backing store is postponed until the cache blocks containing the data are about to be modified/replaced by new content.

A write-back cache is more complex to implement, since it needs to track which of its locations have been written over, and mark them as dirty for later writing to the backing store. The data in these locations are written back to the backing store only when they are evicted from the cache, an effect referred to as a lazy write. For this reason, a read miss in a write-back cache (which requires a block to be replaced by another) will often require two memory accesses to service: one to write the replaced data from the cache back to the store, and then one to retrieve the needed data.

Other policies may also trigger data write-back. The client may make many changes to data in the cache, and then explicitly notify the cache to write back the data. No data is returned on write operations, thus there are two approaches for situations of write-misses:

a. **Write Allocate (Also Called Fetch on Write):** data at the missed-write location is loaded to cache, followed by a write-hit operation. In this approach, write misses are similar to read misses.
b. **No-Write Allocate (Also Called Write-No-Allocate or Write Around):** data at the missed-write location is not loaded to cache, and is written directly to the backing store. In this approach, only the reads are being cached.

Both write-through and write-back policies can use either of these write-miss policies, but usually they are paired in this way:

A write-back cache uses write allocate, hoping for subsequent writes (or even reads) to the same location, which is now cached. A write-through cache uses no-write allocate. Here, subsequent writes have no advantage, since they still need to be written directly to the backing store.

Entities other than the cache may change the data in the backing store, in which case the copy in the cache may become out-of-date or stale. Alternatively, when the client updates the data in the

cache, copies of those data in other caches will become stale. Communication protocols between the cache managers which keep the data consistent are known as coherency protocols.

4. Replacement Algorithms:

A cache algorithm is a detailed list of instructions that directs which items should be discarded in a computing device's cache of information. Examples of cache algorithms include:

a. First In First Out (FIFO):

Using this algorithm the cache behaves in the same way as a FIFO queue. The cache evicts the first block accessed first without any regard to how often or how many times it was accessed before.

b. Last In First Out (LIFO):

Using this algorithm the cache behaves in the exact opposite way as a FIFO queue. The cache evicts the block accessed most recently first without any regard to how often or how many times it was accessed before.

c. Least Frequently Used (LFU):

This cache algorithm uses a counter to keep track of how often an entry is accessed. With the LFU cache algorithm, the entry with the lowest count is removed first. This method isn't used that often, as it does not account for an item that had an initially high access rate and then was not accessed for a long time.

d. Least Recently Used (LRU):

This cache algorithm keeps recently used items near the top of cache. Whenever a new item is accessed, the LRU places it at the top of the cache. When the cache limit has been reached, items that have been accessed less recently will be removed starting from the bottom of the cache. This can be an expensive algorithm to use, as it needs to keep "age bits" that show exactly when the item was accessed. In addition, when a LRU cache algorithm deletes an item, the "age bit" changes on all the other items.

e. Adaptive Replacement Cache (ARC):

Developed at the IBM Almaden Research Center, this cache algorithm keeps track of both LFU and LRU, as well as evicted cache entries to get the best use out of the available cache.

f. Most Recently Used (MRU):

This cache algorithm removes the most recently used items first. A MRU algorithm is good in situations in which the older an item is, the more likely it is to be accessed.

g. Random Replacement (RR):

Randomly selects a candidate item and discards it to make space when necessary. This algorithm does not require keeping any information about the access history. For its simplicity, it has been used in ARM processors. It admits efficient stochastic simulation.

Chapter – 7

Input Output Organization

External Devices:

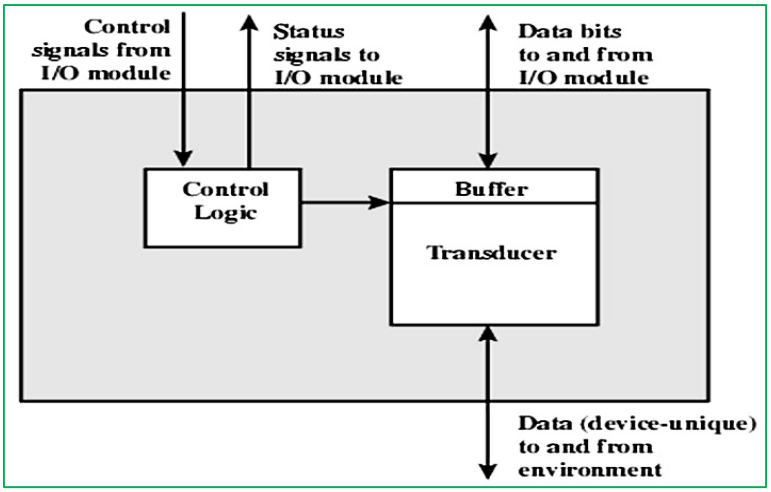


Fig: Block Diagram of an External Device

Input Output operations are accomplished through a wide range of external devices that provides a means of exchanging data between external environment and the computer. An external device is attached to the computer by a link to input/output module. The link is used to exchange control, status and data between input/output module and the external device. An external device connected to input/output module is known as peripheral device or simply a peripheral. The external devices can be classified as:

1. **Human Readable:** Suitable for communicating with the computer user. For example: Screen, Printer, etc.
2. **Machine Readable:** Suitable for communicating with equipment. For example: Magnetic Disk, Memory, etc.
3. **Communications:** Suitable for communicating with remote devices. For example: Modem, Network Interface Card (NIC).

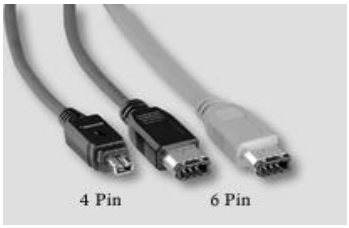
Examples:

1. **SCSI:**

- Acronym for small computer system interface.
- Pronounced "scuzzy," SCSI is a parallel interface standard used by Apple Macintosh computers, PCs, and many UNIX systems for attaching peripheral devices to computers.
- Provide for faster data transmission rates (up to 80 megabytes per second) than standard serial and parallel ports.
- Can attach many devices to a single SCSI port, so that SCSI is really an I/O bus rather than simply an interface.

- However, SCSI rarely in used because each piece of SCSI hardware has its own host adapter, and the software drivers for the device cannot work with an adapter made by someone else.
- Nearly all Apple Macintosh computers, excluding only the earliest Macs and the recent iMac, come with a SCSI port for attaching devices such as disk drives and printers.

2. **FireWire (IEEE 1394):**



- Fire Wire is Apple Computer's version of a standard, IEEE 1394, High Performance Serial Bus, for connecting devices to personal computer.
- Uses for devices that need to transfer high levels of data in real-time, such as video devices.
- Fire Wire provides a single plug-and-socket connection on which up to 63 devices can be attached with data transfer speeds up to 400 Mbps (megabits per second).
- Like USB, it also supports both Plug-and-Play and hot plugging, and also provides power to peripheral devices.

FireWire

3. **USB:**

- USB (Universal Serial Bus) is **a plug-and-play interface between a computer and add-on devices (such as mouse, scanners, and printers).**
- With USB, a new device can be added to computer without having to add an adapter card or even having to turn the computer off.
- USB 1.1 supports a data speed of 12 megabits per second. This speed will accommodate a wide range of devices.
- It is expected to completely replace serial and parallel ports.
- A single USB port can be used to connect up to 127 peripheral devices.

EXTERNAL INTERFACES:

The interface to a peripheral from an input/output module must be tailored to the nature and operation of the peripheral. One major characteristic of the interface is whether it is serial or parallel.

In a parallel interface, there are multiple lines connecting the input/output module and the peripheral, and multiple bits are transferred simultaneously, just as all of the bits of a word are transferred simultaneously over the data bus.

In a serial, there is only one line used to transmit data, and bits must be transmitted one at a time. A parallel interface has traditionally been used for higher speed peripherals, such as tape and disk, while the serial interface has traditionally been used for printers and terminals. With a new generation of high-speed serial interfaces, parallel interfaces are becoming much less common.

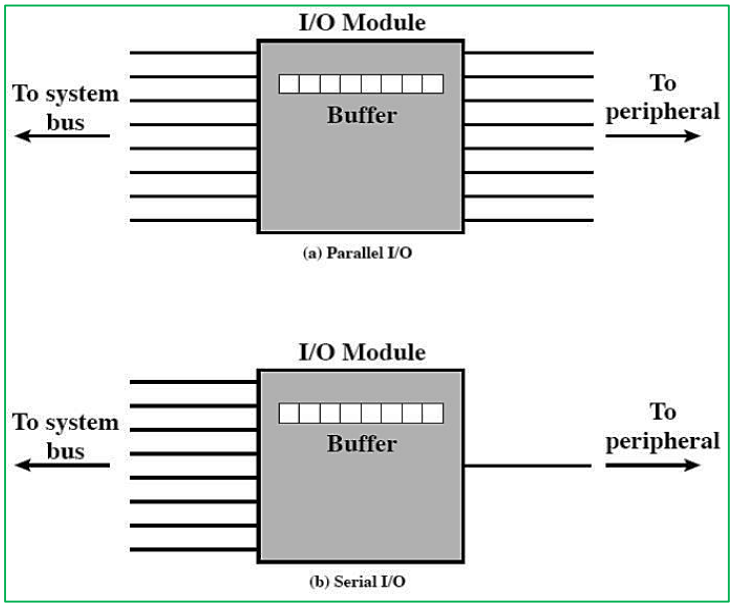


Fig: Parallel and Serial IO

In either case, the input/output module must engages in a dialogue with the peripheral. In general terms, the dialogue for a write operation is as follows:

- The input/output module sends a control signal requesting permission to send data.
- The peripheral acknowledges the request.
- The input/output module transfers data (one word a block depending on the peripheral).
- The peripheral acknowledges receipt of the data.

A read operation proceeds similarly.

Key to the operation of an input/output module is an internal buffer that can store data being passed between the peripheral and the rest of the system. This buffer allows the input/output module to compensate for the differences in speed between the system bus and its external lines.

INPUT OUTPUT MODULE STRUCTURE:

It is an entity with in a computer that is responsible for the control of one or more external devices and for the exchange of data between those devices and main memory or CPU. The input/output memory must have:

a. Interface to CPU and memory
b. Interface to one or more peripheral

The major function or requirement of input/output module falls into following categories:

a. Control and Timing
b. CPU communication
c. Device communication
d. Error detection

Similarly, CPU might involve in sequence of operations like:

a. Check the input/output module device status.

b. Returns device status
c. If ready CPU request data transfer
d. Input/output modules gets data to CPU, etc.

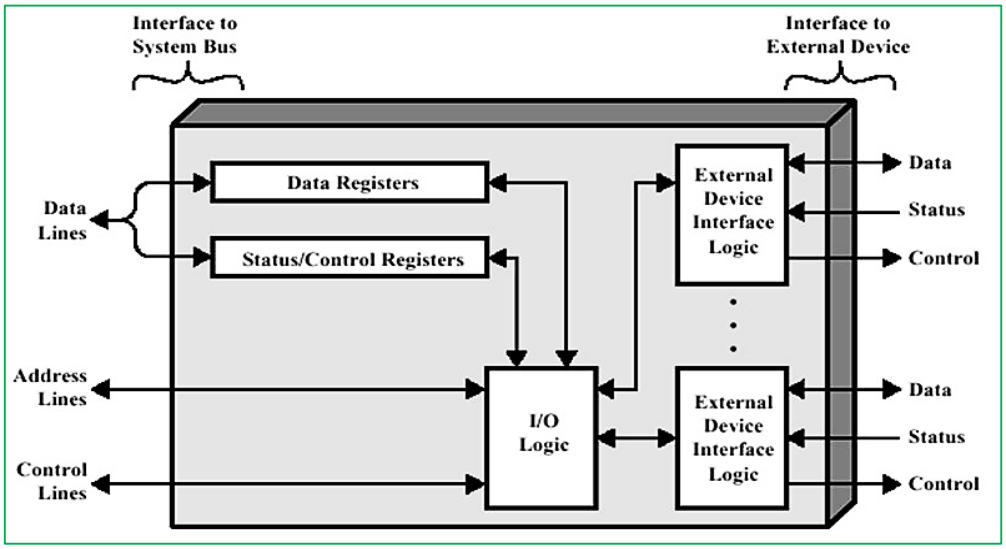


Fig: Block Diagram of Input/output Module

INPUT OUTPUT DATA TRANSFER TECHNIQUES:

There are 3 different types of data transfer techniques:

1. PROGRAMMED INPUT OUTPUT:

With programmed input/output, data are exchanged between CPU and input/output module. The CPU execute a program that gives it direct control of the input/output operation, including sensing the device status, sending a read/write command and transferring the data. When CPU issues a command to input/output module, it must wait until input/output operation is completed.

If CPU is faster than input/output module then there is wastage of CPU time.

Using programmed input/output, the input/output module will perform the requested actions and then set appropriate bits in status register. It is the responsibility of CPU to periodically check the status of the input/output module until it finds that the operation is completed.

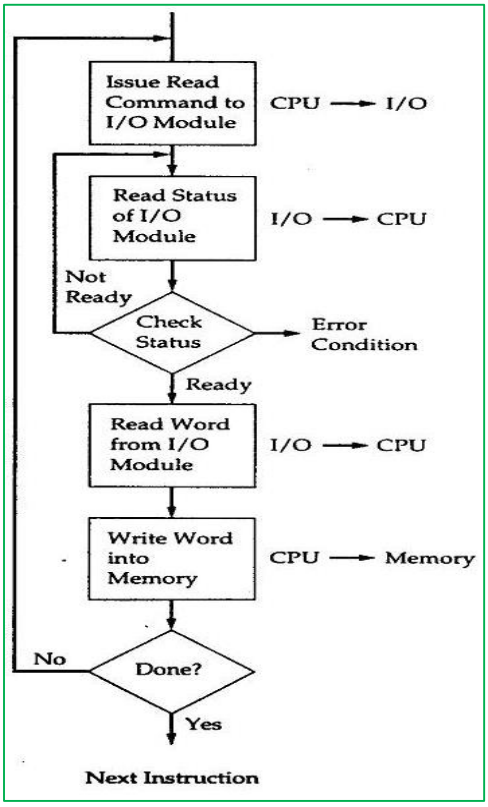


Fig: Programmed IO

2. Interrupt Driven Input Output:

Using programmed control input/output technique there is wastage of the time of CPU, which is overcome by interrupt driven input/output technique. Here, CPU issues read command and when input/output module gets data from peripheral device then it interrupts CPU. After that, CPU request data and input/output module transfer data. The CPU reads data and store it in main memory.

Advantages:

a. Less wastage of the time of CPU.
b. Speed is comparatively faster than programmed input/output technique.

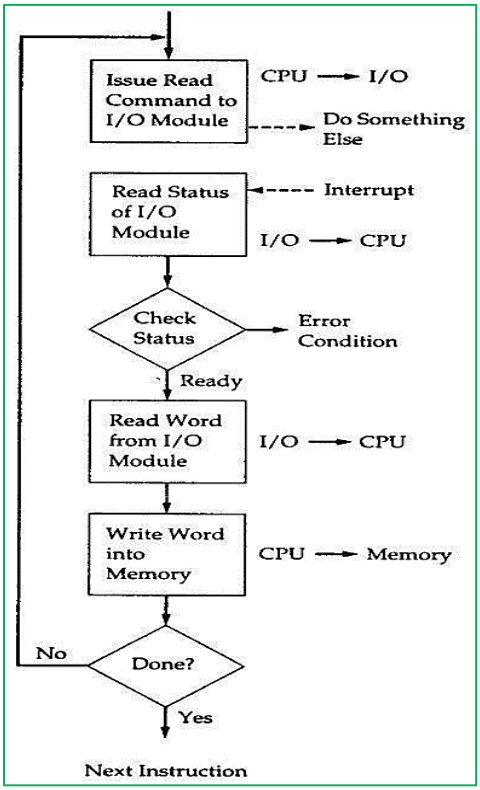


Fig: Interrupt Driven IO

3. Direct Memory Access (DMA):

DMA is the technique, which allows data to be send directly from an attached device to the memory and vice-versa. Usually a specified portion of memory is designated as an area to be used for DMA.

Here, the microprocessor is freed form involvement with the data transfer. Thus, speeding up overall computer operation. As shown in diagram at right side, whether a read or write is requested, there is use of control lines between input/output devices and memory. The address of the input/output device is communicated on the data-lines.

The overall operation for the data transfer is controlled through DMA controller.

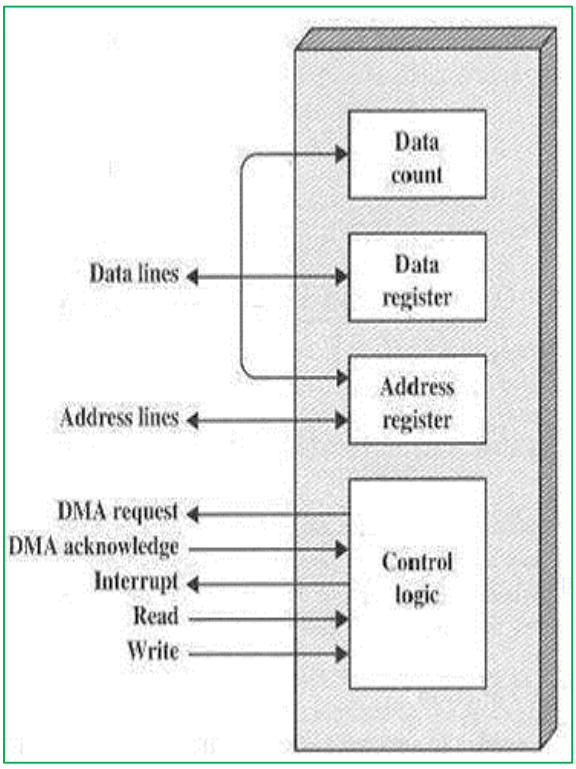


Fig: Block Diagram of DMA

INPUT OUTPUT CHANNELS AND PROCESSOR:

As computer systems has evolved, there is a pattern of complexity and sophistication of individual components. There are different characteristics of input/output channels for accomplishing complicated tasks. It has an ability to execute input/output instructions, which gives it complete control over input/output operations.

Some tasks are:

- Instructions are stored in main memory to be executed by a special purpose processor.
- A selector channel controls multiple high speed devices and at any time.
- The input/output channel selects one device and effects the data transfer.
- Similarly, a multiplexer channel can handle with multiple devices at the same time.
- For low speed devices a byte multiplexer accepts or transmits characters as fast as possible to multiple devices.

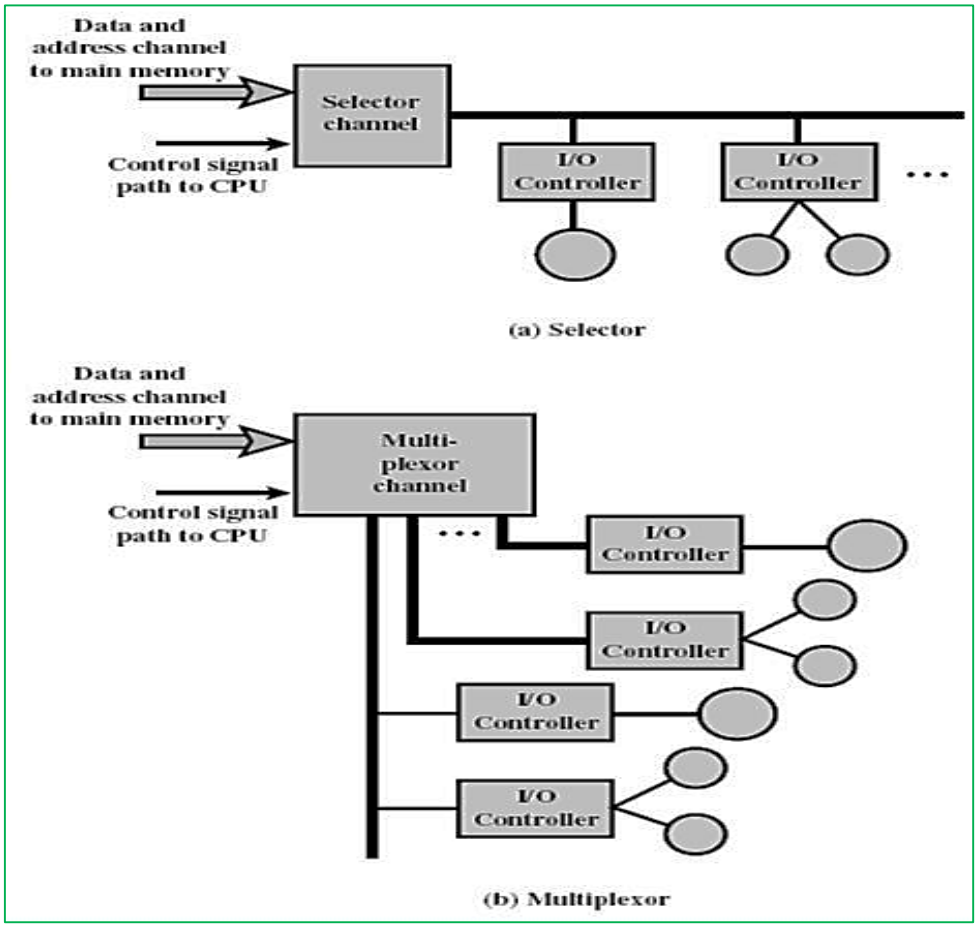


Fig: Input/output Channel Architecture

CHAPTER – 8

REDUCED INSTRUCTION SET COMPUTER (RISC)

INTRODUCTION:

The RISC architecture is dramatic departure from the historical trend in processor architecture. Analysis of RISC architecture brings into focus many of the important issues in computer organization and architecture.

The key elements in design of RISC architecture are:

a. Large number of general purpose registers and the use of compiler technology to optimize the register usage.
b. Limited and simple instruction set.
c. Emphasis on optimizing the instruction pipelining.

FEATURES:

a. Limited and simple instruction
b. Large number of general purpose registers or the use of compiler technology
c. Emphasis on optimizing instruction pipeline
d. Single cycle operation
e. Load or store design

CISC (COMPLEX INSTRUCTION SET COMPUTER):

CISC architecture have a richer instruction sets which include a large number of instructions and more complex instructions.

FEATURES:

a. Complex machine instructions
b. More hardware circuitry
c. More addressing modes.

RISC V/S CISC:

RISC	CISC
Emphasis on software.	Emphasis of hardware.
Instructions of same set with few formats.	Multiple instruction sizes and formats.
Uses more registers.	Uses less registers.
Fewer addressing modes.	More addressing modes.
Instructions take one cycle time that is single clock.	Instructions take varying cycle time that is multi-clock.
Pipelining is easy.	Pipelining is difficult.

INSTRUCTION PIPELINING:

First generation RISC processors achieve execution speed that approach one instruction per system clock cycle. To improve it, two classes of processors evolved to offer execution of multiple instructions per clock cycle i.e. super-scalar and super-pipelined architectures.

A super-scalar architecture replicates each of the pipeline stages so that two or more instructions at the same stage of pipelines. Similarly, super-pipelined architecture is one that makes use of more and finer grained pipeline stages.

All instructions follow the five pipeline stages:

1. Instruction fetch
2. Source operand fetch from register file
3. ALU operation or data operand address generation
4. Data memory reference
5. Write back into register file

The limitation of super-scalar is dependencies between instructions in different pipelines can slow down the system. With super-scaling there is overhead associated with transferring instructions form one stage to another.

PIPELINING NUMERICAL:

$$Speed\ up\ ration = \frac{time\ to\ execute\ 'n'\ instructionsin\ non\ pipelined\ design}{time\ to\ execute\ 'n'\ instructions\ in\ pipelined\ design}$$

$$= \frac{k * n * t}{k * n + (n-1) * t}$$

$$= \frac{k * n}{k + (n-1)}$$

Where,

- k = number of segments
- n = number of tasks
- t = time to process a sub-operation

EXAMPLE:

Assume that pipeline has k = 8 segments and execute n = 120 tasks in sequence. Let the time taken to process a sub-operation in each segment is 20 second calculate the speed of ration in the pipeline.

Solution,

$$speed\ up\ ration = \frac{k * n * t}{k * t + (n-1) * t}$$

$$= \frac{8 * 120 * 20}{8 * 20 + (120-1) * 20}$$

$$= \frac{8 * 120 * 20}{20(8 + 119)}$$

$$= \frac{8 * 120}{8 + 119}$$

$$= \frac{960}{127}$$

$$= 7.56$$

RISC PIPELINING:

Define two phase of execution for register based instruction:

- I: Instruction Fetch
- E: Execute
  - ALU operation with register input and output.

For load and store there will be three phases:

- I: Instruction fetch
- E: Execute
  - Calculate memory address
- D: Memory
  - Register to memory or memory to register operation

Load rA ← M	I	E	D										
Load rB ← M				I	E	D							
Add rC ← rA + rB							I	E					
Store M ← rC									I	E	D		
Branch X												I	E

Fig: Sequential Execution

Load rA ← M	I	E	D								
Load rB ← M		I		E	D						
Add rC ← rA + rB				I		E					
Store M ← rC						I	E	D			
Branch X							I		E		
NOOP										I	E

Fig: Two Stage Pipelining

CONFLICTS OR HAZARDS IN INSTRUCTION PIPELINING AND THEIR SOLUTIONS:

Pipelining hazards are the problems that may be created due to the use of pipelining. Different types of hazards in pipelining are:

1. Structural Hazards:

Attempt to use the same resource by two or more instructions. For example: Access at same time by memory stage.

Solution:

a. Delay the second access by one clock cycle.
b. Provide separate memory for instructions and data.

2. Control Hazards:

Attempt to make branching decision before branch condition is evaluated.

Solution:

a. Predict: Assume an outcome and continue fetching (undo) if prediction is wrong.
b. Delayed branch:
c. Stall: Stop loading instructions until result is available.

3. Data Hazards:

Attempt to use the data before it is ready.

Solution:

a. Read after write
b. Write after read
c. Write after write (out dependencies)

Register Window:

1. Use of large set of registers decreases the need to access memory.
2. Design task is to organize registers in such a way that this goal is achieved.
3. Multiple small set of registers are used, each assigned to a different procedure.
4. A procedure call automatically switches the processor to use a different fixed size window of registers.
5. At any time only one window of register is visible and addressable as if it were the only set of registers.
6. The window is divided into three fixed size area i.e. parameter register, local register, and temporary register.
7. Parameter register hold the parameters passed down from the procedure that call the current procedure and holds the result to be passed as a backup.
8. Local registers are used for local variables as assigned by the compiler.
9. Temporary registers are used to exchange parameters and result with the next lower level. The overlap permits parameters to be passed without actual movement of data.

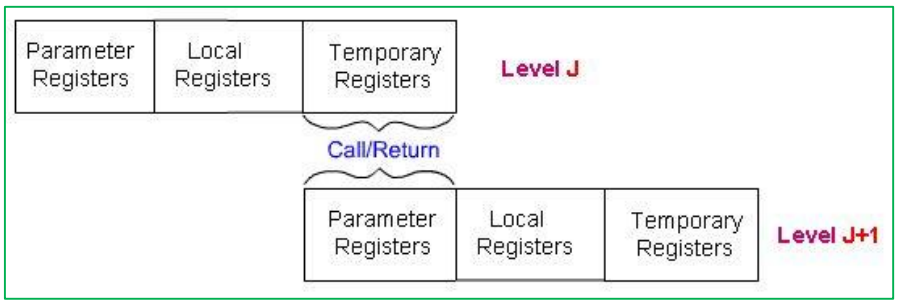


Fig: Overlapping Register Window

REGISTER RENAMING:

- ☑ When instructions are issued in sequence and complete in sequence, it is possible to specify the contents of each register at each point in the execution.
- ☑ When out of order techniques are used, the values of registers cannot be fully known at each point.
- ☑ Due to this effect, values are in conflict for the use of registers and processor must resolve these conflicts by the process of pipelining. Anti-dependencies and out dependencies are the examples of storage conflicts.
- ☑ Multiple instructions compete for the use of same register location generating pipeline constraints that retards performance.
- ☑ The method of coping with these type of conflict is duplication of resources called register renaming.

Chapter – 9

Introduction to Parallel Processing

Parallel Processing:

Parallel processing is a method used to improve performance in a computer system. When a system processes two different instructions simultaneously then it is performing parallel processing.

a. Parallelism in Uniprocessor System:

Uniprocessor system can perform two or more task simultaneously. The tasks are not related to each other so a system that processes two different instructions simultaneously could be considered to perform parallel processing.

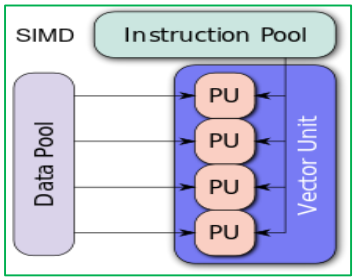
b. Parallelism in Multi-processor System:

Parallel processing system achieve parallelism by having more than one processor performing tasks simultaneously. Since, multi-processor systems are more complicated than uniprocessor system, there are many different ways to organize processors and memory. So, researcher Flynn purposed a classification based on the flow of instructions and data with in a computer called Flynn's classification.

Flynn's Classification:

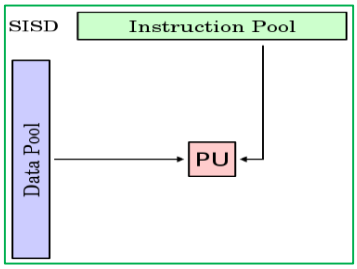
It is based on instructions and data processing:

1. SIMD (Single Instruction Multiple Data):



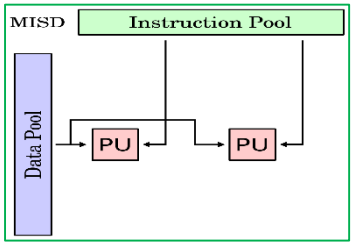
SIMD machine executes a single instruction on multiple data values simultaneously using multi-processors. Since, there is only one instruction each processor does not have to fetch and decode each instruction.

2. SISD (Single Instruction Single Data):



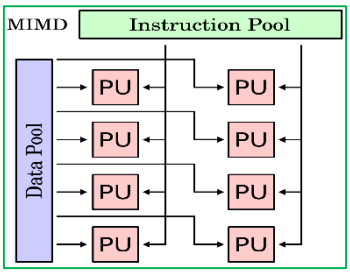
SISD machine executes a single instruction on individual data values using single processor, even if the processor incorporates internal parallelism such as an instruction pipelines, the computer would still be classified as SISD.

3. MISD (Multiple Instruction Single Data):



This classification is not practical to implement. So, number of significant MISD computer are not implemented.

4. MIMD (Multiple Instruction Multiple Data):



System referred to as multi-processor or multi-computers are usually MIMD. It may execute multiple instructions simultaneously so, each processor must include its own control unit. MIMD machines are well-suited for general purpose use.

The multiprocessors are further classified into two groups depending on the way their memory is organized. **The processors with shared memory are called tightly coupled or shared memory processors.** The information in these processors is shared through the common memory. Each of the processors can also have their local memory too. **The other class of multiprocessors is loosely coupled or distributed memory multi-processors.** In this, each processor have their own private memory, and they share information with each other through interconnection switching scheme or message passing.

The principal characteristic of a multiprocessor is its ability to share a set of main memory and some I/O devices. This sharing is possible through some physical connections between them called the interconnection structures.

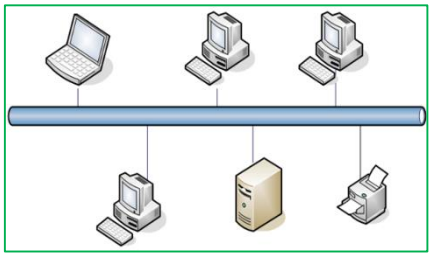
Interconnection Structures in Multiprocessors:

Although topology differs greatly, standard metrics or parameters such as diameter and bandwidth are used to quantify them.

- **Diameter:** The maximum distance between two processors in the computer system.
- **Bandwidth:** The capacity of a communication link.

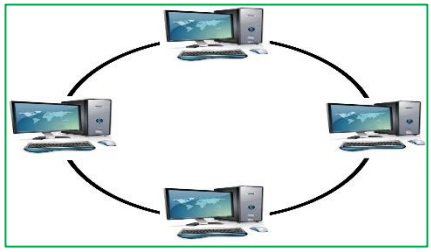
Types:

1. Shared Bus Topology:



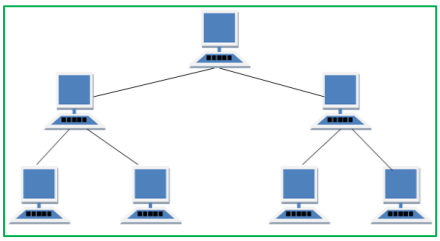
In this topology, processors communicate with each other exclusively through this bus. However, the bus can only handle one data transmission at a time. In most shared buses, processors directly communicate with their own local memory.

2. Ring Topology:



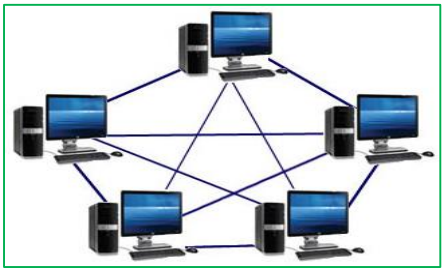
The ring topology uses direct communication between processors instead of a shared bus. This allows all communication links to be active simultaneously. Data may have to travel through several processors to reach destination.

3. Tree Topology:



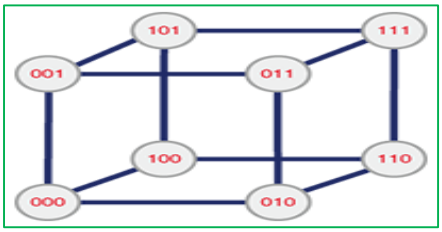
Like the ring, it uses direct communication between processors each having two or more connections. There is only one unique path between each pair of processors.

4. Mesh Topology:



In mesh topology, every processors are connected with one another, allowing for most transmissions to be distributed, even if one of the connections go down. A mesh topology can be a **full mesh topology** or a **partially connected mesh topology**.

5. Hyper-cube Topology:



The hyper-cube is a multi-dimensional mesh topology. Here, each processor connect to all other processors whose binary values differ by one bit.

CACHE COHERENCE:

When multiple processors with separate caches share a common memory, it is necessary to keep the caches in a state of coherence by ensuring that any shared operand that is changed in any cache is changed throughout the entire system to maintain data consistency. This is done in either of two ways:

1. DIRECTORY BASED PROTOCOL:

In a directory based system the data being shared is placed in a common directory that maintains the coherence between caches. The directory acts as a filter through which the processor must ask permission to load an entry from the primary memory to its cache. When an entry is changed the directory either updates or invalidates the other caches with that entry.

2. SNOOP PROTOCOL:

In a snooping system, all caches on the bus monitor (or snoop) the bus to determine, if they have a copy of the block of data that is requested on the bus. Every cache has a copy of the sharing status of every block of physical memory it has.

Cache misses and memory traffic due to shared data blocks limit the performance of parallel computing in multiprocessor computers or systems. Cache coherence aims to solve the problem associated with sharing data.

INTRODUCTION TO VECTOR PROCESSING AND ARRAY PROCESSORS:

1. VECTOR PROCESSING:

There is a class of computational problems that are beyond the capabilities of the conventional computer. These are characterized by the fact that they require vast number of computation and it take a conventional computers days or even weeks to complete. Computers with vector processing are able to handle such instruction and they have application in following fields:

- Long Range Weather Forecasting
- Petroleum Exploration
- Seismic Data Analysis
- Medical Diagnosis
- Aerodynamics And Space Simulation
- Artificial Intelligence And Expert System
- Mapping The Human Genome
- Image Processing

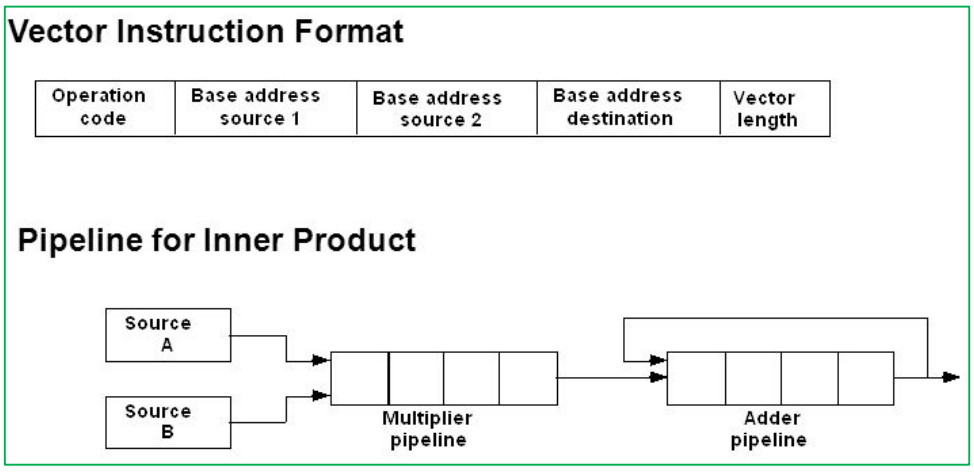
Vector Operation:

A vector 'V' of length 'n' is represented as row vector by V = [V_1, V_2, V_3, … V_n]. The element Vi of vector 'V' is written as V (I) and the index 'I' refers to a memory address or register where the number is stored.

Let us consider the program in assembly language that two vectors A and B of length 100 and put the result in vector C.

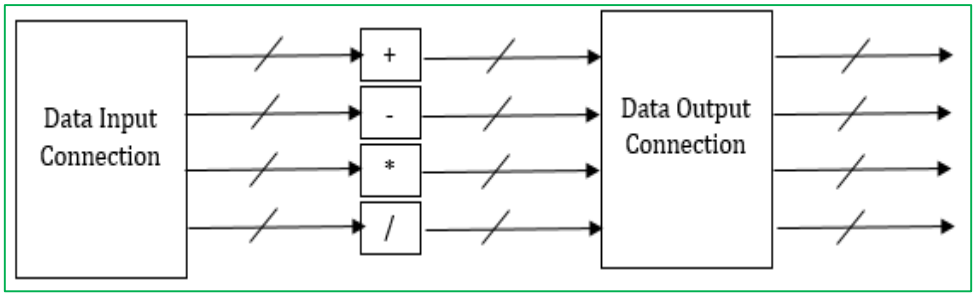
```
        Initialize I = 0
20      Read A (I)
        Read B (I)
        Store C (I) = A (I) + B (I)
        Increment I = I + 1
        If I <= 100 go to 20
        Continue
```

A computer capable of vector processing eliminates the overhead associated with the time it takes to fetch and execute the instructions in the program loop. It allows operations to be specified with a single vector instruction of the form: C (1:100) = A (1:100) + B (1:100)



Vector Arithmetic Unit:

Vector arithmetic unit is used to perform different arithmetic operations in parallel. A vector arithmetic unit contains multiple functional units. Some performs addition, other subtraction and in similar manner other performs different operations.



To add two numbers the control unit routes these value, to an adder unit.

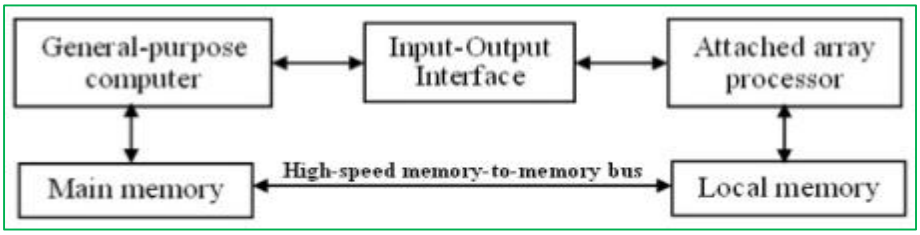
For the operations A ←B + C, D ← E – F, the CPU operations route B and C to adder and E and F to subtractor. This allows CPU to execute both instructions simultaneously.

2. Array Processor:

An array processor is a processor that performs the computations on large arrays of data. There are two different types of array processor:

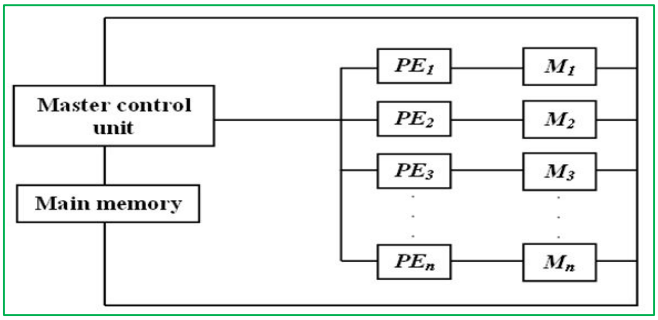
a. Attached Array Processor:

It is designed as a peripheral for a conventional host computer. Its purpose is to enhance the performance of the computer by providing vector processing. It achieves high performance by means of parallel processing with multiple functional units.



b. SIMD Array Processor:

It is processor which consists of multiple processing unit operating in parallel. The processing units are synchronized to perform the same task under control of common control unit. Each processor elements (PE) includes an ALU, a floating point arithmetic unit and working register.



INTRODUCTION TO MULTITHREADED ARCHITECTURE:

Multithreading is the ability of the central processing unit or a single core in a multi-core processor to execute multiple processes or threads concurrently and appropriately supported by operating system. This approach differs from multi-processing as threads have to share the resources of a single or multiple cores.

Multi-threading aims to increase utilization of a single core by using thread level as well as instruction level parallelism.

Merits:

- If a thread gets a lots of cache misses then other threads can continue taking the advantage of unused computing structure.
- If several threads work on the same set of data then they can actually share their cache leading to better cache uses.

Demerits:

- Multiple threads can interfere with each other while sharing hardware resources.

CHAPTER – 10

MULTICORE COMPUTERS

INTRODUCTION:

A multicore computer also known as chip multiprocessor combines two or more processors (cores) on a single piece of silicon.

HARDWARE PERFORMANCE ISSUES:

Microprocessor systems have experienced a steady, exponential increase in execution performance.

1. INCREASE IN PARALLELISM:

The organizational changes in processor design has been primarily focused on increasing instruction level parallelism so, more work can be done in each clock cycles.

2. PIPELINING:

Individual instructions are executed through a pipeline of stages so that while one instruction is executing in one stage of pipeline, another in another stage.

3. SUPERSCALAR:

Multiple pipelines are constructed by replicating execution resources. This enables parallel execution of instructions in pipeline.

4. SIMULTANEOUS THREADING:

Register banks are replicated so that multiple threads can share the use of pipeline resources.

5. POWER CONSUMPTION:

As the number of transistors per chip rise, power requirements grow, one way to control power density is to use more of the chip area for cache memory.

SOFTWARE PERFORMANCE ISSUES:

1. SOFTWARE IN MULTICORE:

The potential performance benefits of a multicore organization depends on the ability to effectively exploit parallel resources available to the application. The Amdahl's law state that,

$$Speed\ up = \frac{time\ to\ execute\ program\ on\ single\ processor}{time\ to\ execute\ program\ on\ N\ parallel\ processors}$$

$$= \frac{1}{(1-f)+\frac{f}{N}}$$

The fraction (1 - f) involves code that is inherently serial and fraction 'f' that involves code that is infinitely parallel.

MULTICORE ORGANIZATION:

The main variable in multicore organization are as follows:

- The number of core processors on the chip.
- The number of levels of cache memory.
- The amount of cache memory that is shared.

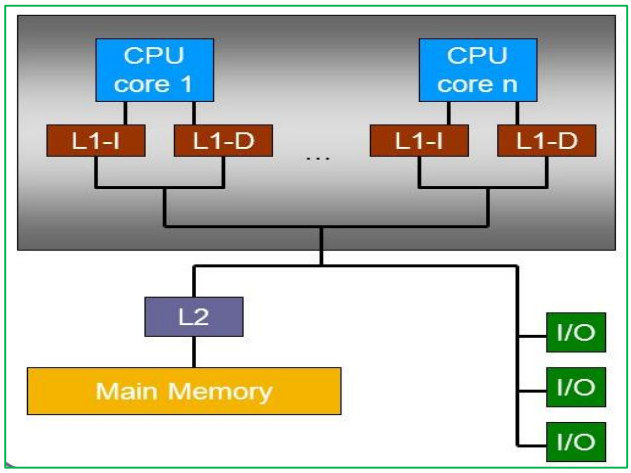


Fig (a): Dedicated L1 Cache

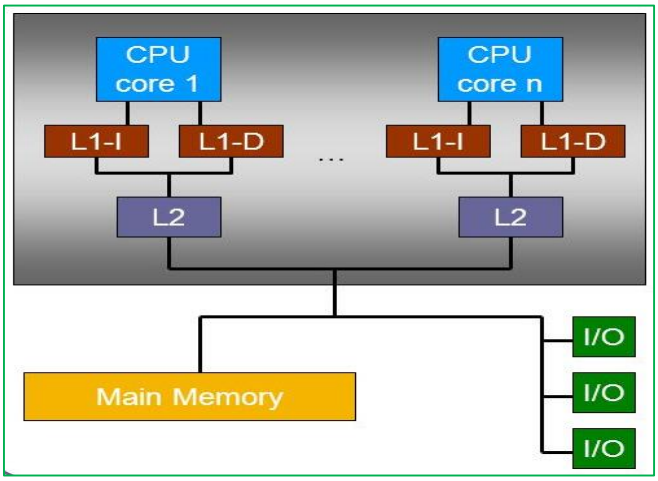


Fig (b): Dedicated L2 Cache

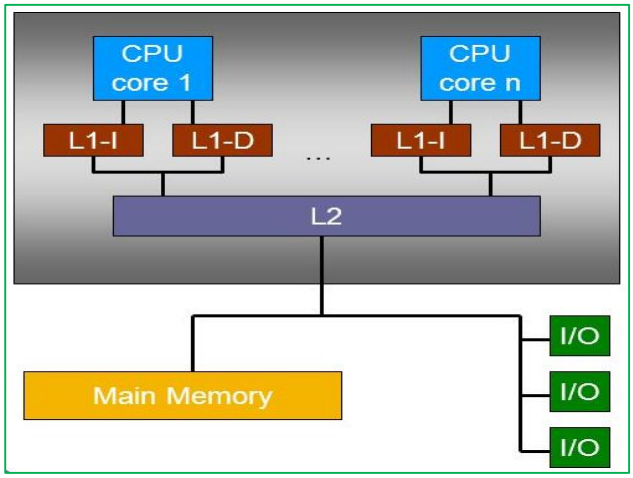


Fig (c): Shared L2 Cache

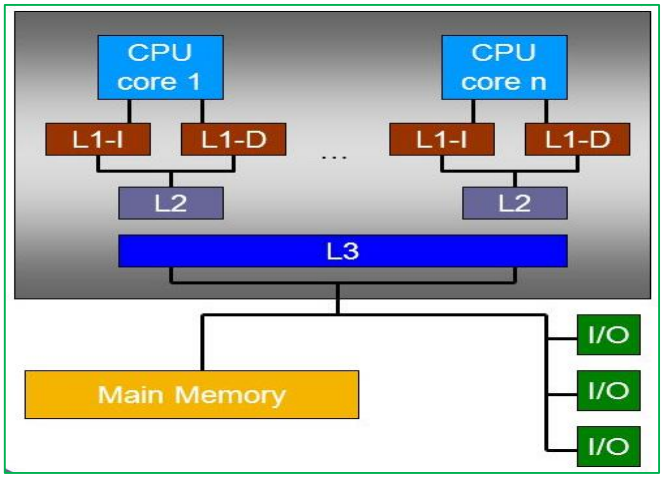


Fig (d): Shared L3 Cache

- D = Data
- I = Instruction

Figures above shows four general organizations for multicore systems.

Figure (a) is an organization found in some of the earlier multicore computer chips and still found in embedded chips. Here, only on chip is L1 with each core having its dedicated L1 cache. L1 cache is divided into instruction and data caches. For example: ARM Processors

Figure (b) is also one in which there is no on chip cache sharing there is enough area available on the chip to allow for L2 cache. For example: AMD Processor.

Figure (c) shows a similar allocation of chip space to memory but with the use of shared L2 cache. For example: Intel Core Duo

Figure (d) shows that the amount of cache memory available on the chip continuous to grow having a shared L3 cache with dedicated L1 and L2 for each core processor. For example: Intel Core i7.

DUAL CORE AND QUAD CORE PROCESSOR:

1. Dual Core Processor:

- CPU that includes two complete execution cores.
- Combines two processors and their cache controllers in a single chip.
- Operating system has sufficient resources to handle intensive tasks in parallel.
- Example: AMD Processor, Intel Core 2 Duo, etc.

2. Quad Core Processor:

- Has four independent units i.e. cores.
- Individual core can run multiple instructions at that same time.
- Has more processing speed.
- Example: Intel Core i5

POWER EFFICIENT PROCESSORS:

Power management has become a major issue in the design of multi-core chips. There are many negative effects that result from increasing power consumption such as unstable thermal properties of the die and hence affecting the system performance which makes power consumption issue sometimes more important than speed. An important observation is that threads running on different cores do not need the same power all time to execute at high performance. There are some waiting times due to memory read/write operations for example which require saving unnecessary processing power. So, to achieve a good balance between scalar performance/throughput performance and power it is essentially required to dynamically vary the amount of power used for processing according to temporal analysis of the code needs.

Developed power management techniques can be classified into two main categories: reactive and predictive.

In reactive techniques, the technique reacts to performance changes in the workload. In other words, a workload may initially have states that need high performance, others of I/O waits and low performance. When the state of the workload changes, the technique reacts to that change accordingly. However, there might be some lag between workload phase changes and power adaptation changes which may lead to states of either in-efficient energy consumption or performance degradation.

On the other hand, predictive techniques, for example, overcome this issue. Those techniques predict phase changes in the workload before they happen, and hence act immediately before a

program phase changes. That leads to optimal energy-saving and performance results. However, there is no workload that can be fully predicted, so reactive techniques are used for portions that cannot be predicted (which is usually more than 60% of the entire workload). So, reactive techniques are inevitable to use and consequently we concentrate in this study on those techniques.

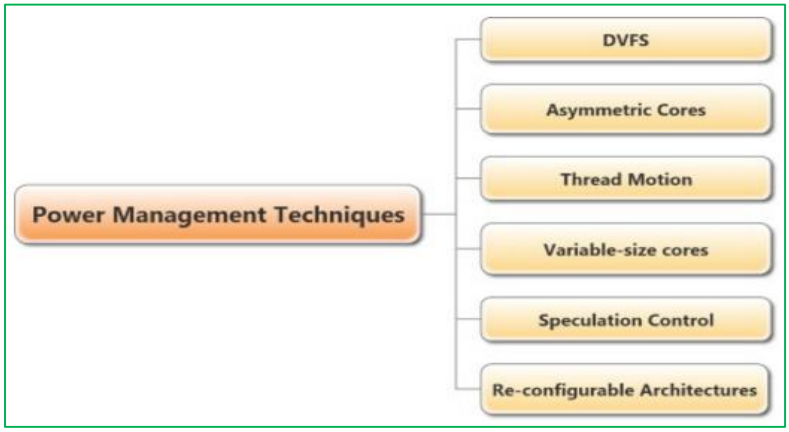


Fig: Power Management Techniques