Chapter 3 Central Processing Unit

3.1 CPU Organization and Structure

The part of the computer that performs the bulk of data processing operations is called the central processing unit (CPU). The CPU is made of three major parts as in figure:

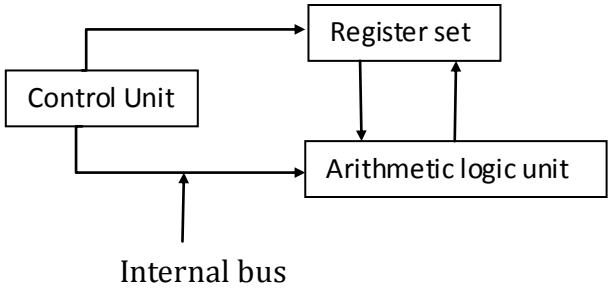


Figure 3.1: Major Components of CPU

The register set stores immediate data used during the execution of the instructions.

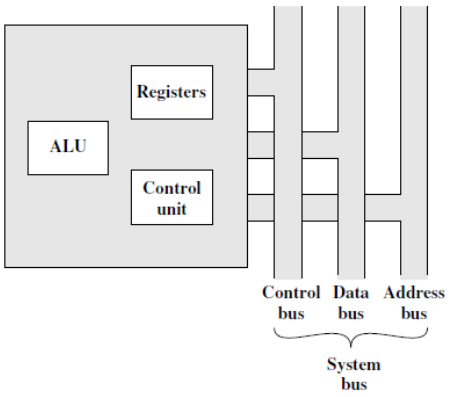
Control unit: It supervises the transfer of information among the registers and instructs the ALU (Arithmetic and Logic Unit) as to which operations to perform.

ALU (Arithmetic and logic Unit): Performs all the arithmetic and logical operations.

A computer consists of set of components or modules of three basic types (processor, memory, I/O) that communicate with each other. The collection of paths connecting the various modules is called the interconnection structure. The design of this structure will depend on the exchange of data and information that must be made among the modules.

Bus interconnection

A bus is a communication pathway connecting two or more devices, which is shared transmission medium. Multiple devices connect to the bus and a signal transmitted by one device is available for reception by all other devices attached to the bus. If two devices transmit during the same time period their signals will overlap and become garbled, thus only one device can transmit successfully at a time. Typically a bus consists of multiple communication pathways or lines, each lines is capable of transmitting signals representing binary 1 or 0.



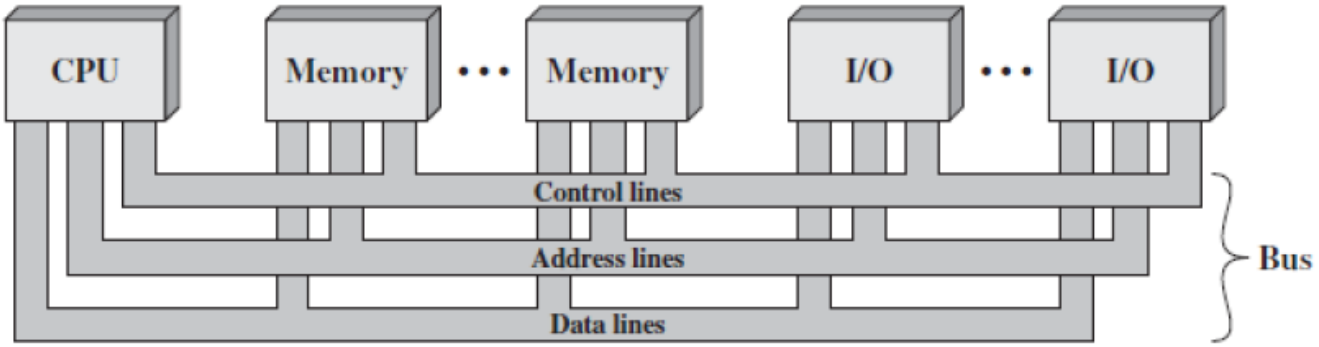


Figure 3.2: Bus interconnection Scheme

- Data line: provide a path for moving data among system modules.
- Address line: used to designate the source or destination of the data on data bus.
- Control lines: used to control the access to and use of data and address lines.

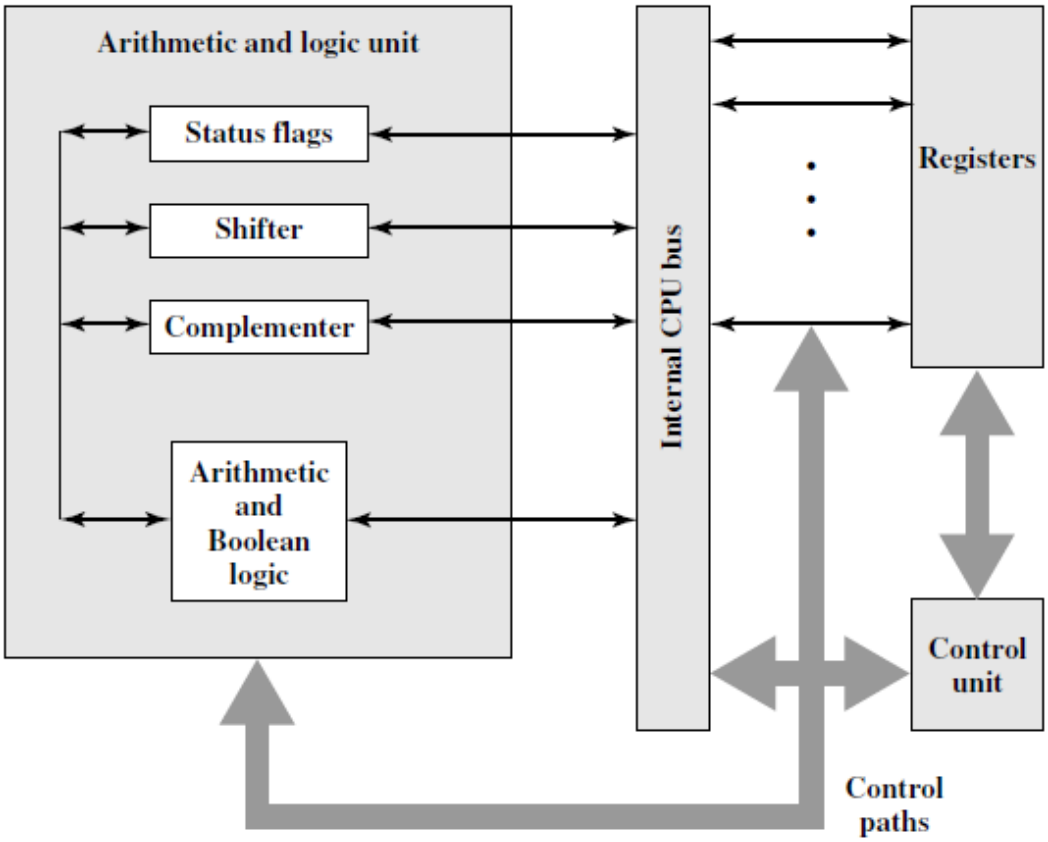


Figure 3.3: Internal structure of the CPU

3.2 Register organization

Register are faster, smaller and more expensive memory that function as a level of memory above main memory and cache. The two categories to register are:

a) **User-visible registers:** a user visible register is one that may be referenced by means of the machine language that the processor executes.
b) **Control and status registers:** Used by the control unit to control the operation of the processor and by privileged, operating system programs to control the execution of programs.

a) **User-visible registers:** a user visible register is one that may be referenced by means of the machine language that the processor executes. We can further categorize as follows:

1. General purpose
2. Data
3. Address
4. Condition codes

1. *General-purpose registers* can be assigned to a variety of functions by the programmer. Sometimes their use within the instruction set is orthogonal to the operation. i.e, any general-purpose register can contain the operand for any op-code. This provides true general-purpose register use. Often, however, there are restrictions. For example, there may be dedicated registers for floating-point and stack operations. In some cases, general-purpose registers can be used for addressing functions (e.g., register indirect, displacement). In other cases, there is a partial or clean separation between data registers and address registers.
2. *Data register:* Data registers may be used only to hold data and cannot be employed in the calculation of an operand address.
3. *Address registers* may themselves be somewhat general purpose, or they may be devoted to a particular addressing mode. Examples include the following:
   i. *Segment pointers:* In a machine with segmented addressing, a segment register holds the address of the base of the segment. There may be multiple registers: for example, one for the operating system and one for the current process.
   ii. *Index registers:* These are used for indexed addressing and may be auto indexed.
   iii. *Stack pointer:* If there is user-visible stack addressing, then typically there is a dedicated register that points to the top of the stack. This allows implicit addressing; i.e, push, pop, and other stack instructions need not contain an explicit stack operand.
   iv. *Conditional codes (also referred to as flags):* Condition codes are bits set by the processor hardware as the result of operations. For example, an arithmetic operation may produce a positive, negative, zero, or overflow result. In addition to the result itself being stored in a register or memory, a condition code is also set. The code may subsequently be tested as part of a conditional branch operation. Condition code bits are collected into one or more registers. Usually, they form part of a control register. Generally, machine instructions allow these bits to be read by implicit reference, but the programmer cannot alter them.

b) Control and status register :

There are a variety of processor registers that are employed to control the operation of the processor. Most of these, on most machines, are not visible to the user. Some of them may be visible to machine instructions executed in a control or operating system mode. Of course, different machines will have different register organizations and use different terminology. Four register are essential to instruction execution:

1. **Program counter (PC):** Contains the address of an instruction to be fetched.
2. **Instruction register (IR):** Contains the instruction most recently fetched.
3. **Memory Address Register (MAR):** Contains the address of a location in memory.
4. **Memory Buffer Register (MBR):** Contains a word of data to be written to memory or the word most recently read.

Many processor designs include a register or set of registers, often known as the ***program status word* (PSW),** that contain status information. The PSW typically contains condition codes plus other status information. Common fields or flags include the following:

• **Sign:** Contains the sign bit of the result of the last arithmetic operation.

• **Zero:** Set when the result is 0.

• **Carry:** Set if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high-order bit. Used for multiword arithmetic operations.

• **Equal:** Set if a logical compare result is equality.

• **Overflow:** Used to indicate arithmetic overflow.

• **Interrupt Enable/Disable:** Used to enable or disable interrupts.

• **Supervisor:** Indicates whether the processor is executing in supervisor or user mode. Certain privileged instructions can be executed only in supervisor mode, and certain areas of memory can be accessed only in supervisor mode.

Note: see example of microprocessor register organization MC68000 & intel 8086 from William Stallings book.

3.3 Instruction cycle

The processing required for a single instruction is called an *instruction cycle*. The instruction cycle can be simply depicted in Figure 3.3. The two steps are referred to as the *fetch cycle* and the *execute cycle*. Program execution halts only if the machine is turned off, some sort of unrecoverable error occurs, or a program instruction that halts the computer is encountered.

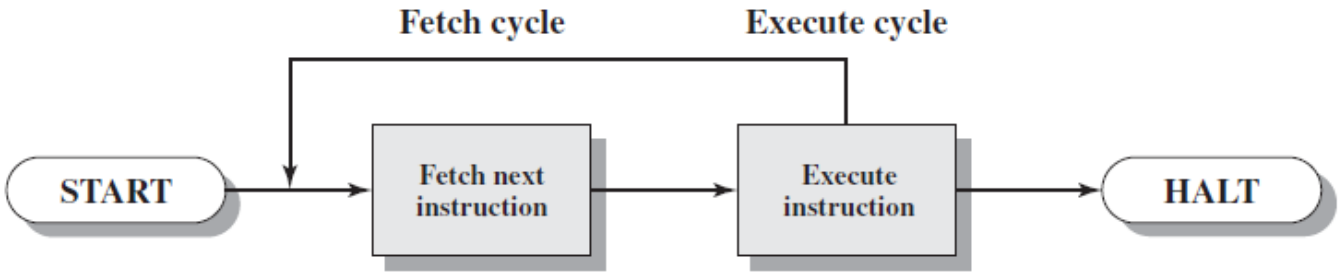


Figure 3.4 Basic instruction cycle

An instruction cycle includes the following stages:

• **Fetch:** Read the next instruction from memory into the processor.

• **Execute:** Interpret the op-code and perform the indicated operation.

• **Interrupt:** If interrupts are enabled and an interrupt has occurred, save the current process state and service the interrupt. We are now in a position to elaborate somewhat on the instruction cycle. First, we must introduce one additional stage, known as the indirect cycle.

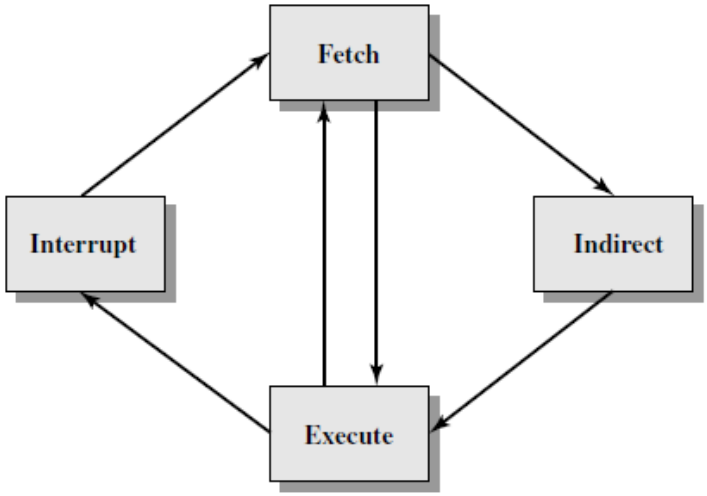


Figure 3.5 instruction cycle

The execution of an instruction may involve one or more operands in memory, each of which requires a memory access. Further, if indirect addressing is used, then additional memory accesses are required. We can think of the fetching of indirect addresses as one more instruction stages. The result is shown in Figure 3.5.The main line of activity consists of alternating instruction fetch and instruction execution activities. After an instruction is fetched, it is examined to determine if any indirect addressing is involved. If so, the required operands are fetched using indirect addressing.

Instruction cycle state diagram

The figure 3.6 shows the state diagram of instruction cycle. For any given instruction cycle some states may be null and others may be visited more than once. The different states may be described as:

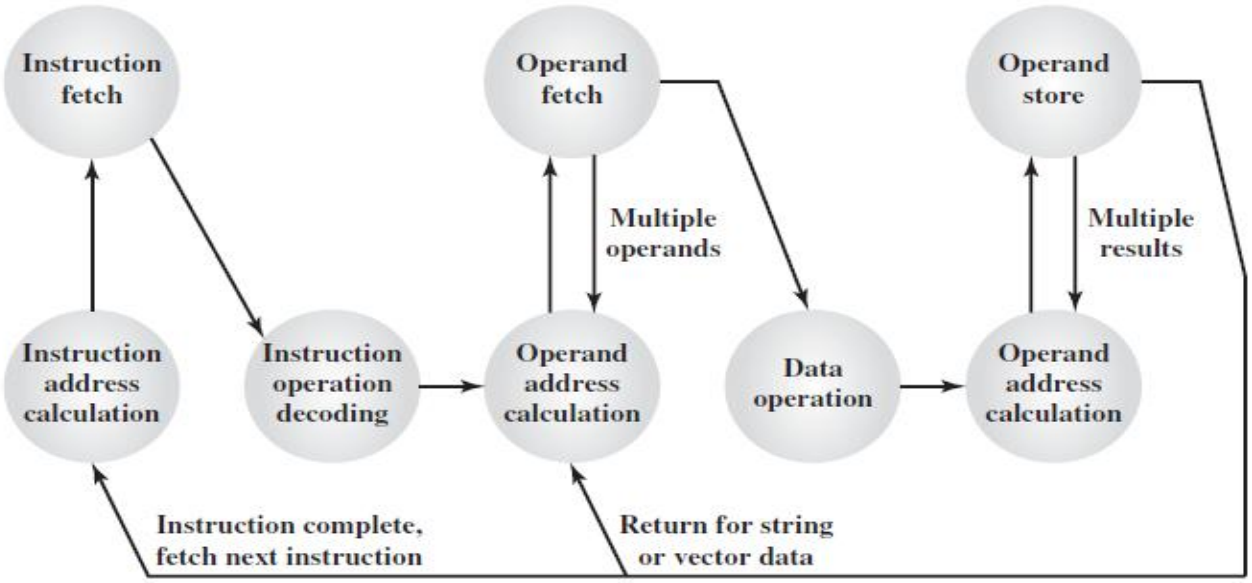


Figure 3.6: Instruction cycle state diagram

Instruction address calculation (iac): Determine the address of the next instruction to be executed. Usually, this involves adding a fixed number to the address of the previous instruction. For example, if each instruction is 16 bits long and memory is organized into 16-bit words, then add 1 to the previous address. If, instead, memory is organized as individually addressable 8-bit bytes, then add 2 to the previous address.

• **Instruction fetch (if):** Read instruction from its memory location into the processor.
• **Instruction operation decoding (iod):** Analyze instruction to determine type of operation to be performed and operand(s) to be used.
• **Operand address calculation (oac):** If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.
• **Operand fetch (of):** Fetch the operand from memory or read it in from I/O.
• **Data operation (do):** Perform the operation indicated in the instruction.
• **Operand store (os):** Write the result into memory or out to I/O.

States in the upper part of Figure 3.5 involve an exchange between the processor and either memory or an I/O module. States in the lower part of the diagram involve only internal processor operations. The oac state appears twice, because an instruction may involve a read, a write, or both. Finally, on some machines, a single instruction can specify an operation to be performed on a vector (one-dimensional array) of numbers or a string (one-dimensional array) of characters. As Figure 3.6 indicates, this would involve repetitive operand fetch and/or store operations.

Interrupts

Interrupts is an asynchronous service request from hardware or software to CPU, interrupts makes CPU execution of its normal operation to pause and then to service external device. The processor and OS are responsible for recognizing an interrupt, suspending the user program servicing the interrupt and then resuming the user program. The instruction cycle with interrupts is as shown in figure.

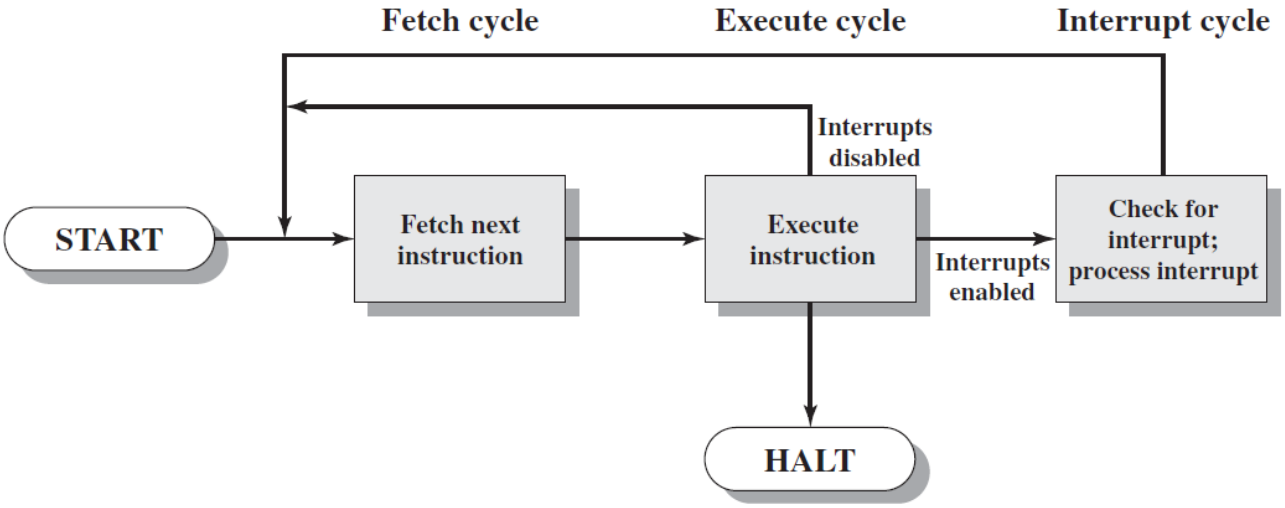


Figure 3.7 Instruction cycle with interrupt.

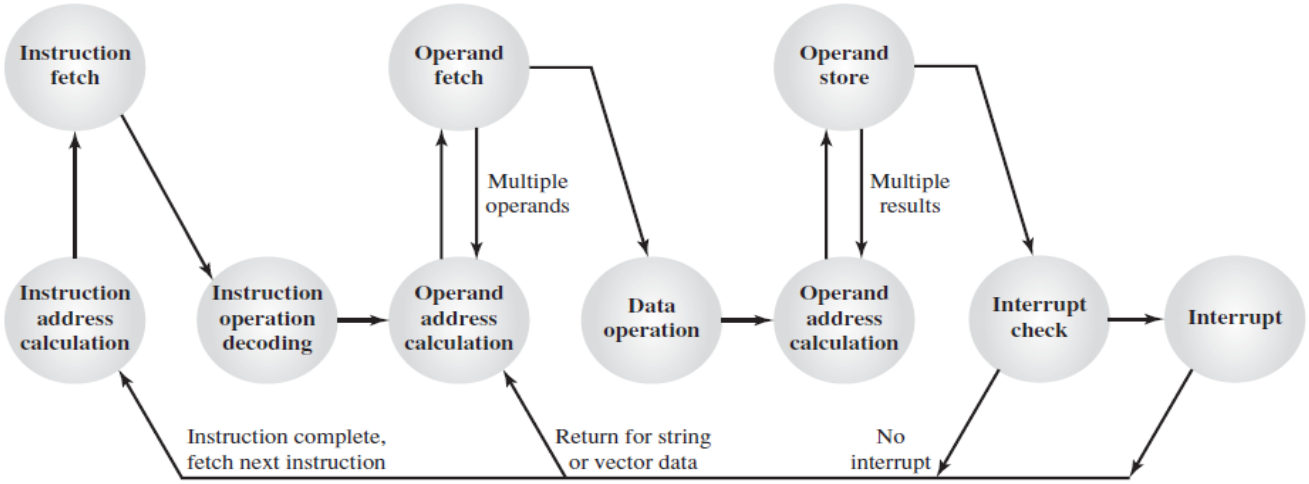


Figure 3.8 Instruction cycle state diagram with interrupt

In the interrupt cycle, the processor checks to see if any interrupts have occurred, indicated by the presence of an interrupt signal. If no interrupts are pending, the processor proceeds to the fetch cycle and fetches the next instruction of the current program. If an interrupt is pending, the processor does the following:

- It suspends execution of the current program being executed and saves its context. This means saving the address of the next instruction to be executed (current contents of the program counter) and any other data relevant to the processor's current activity.
- It sets the program counter to the starting address of an *interrupt handler* routine.

The processor now proceeds to the fetch cycle and fetches the first instruction in the interrupt handler program, which will service the interrupt. The interrupt handler program is generally part of the operating system. Typically, this program determines the nature of the interrupt and performs whatever actions are needed.

3.4 **The arithmetic and logic unit**

The ALU is that part of the computer that actually performs arithmetic and logical operations on data. ALU is a multi operation combinational logic digital function i.e. can perform set of arithmetic and logic of operations. Figure shows the block diagram of 4 bit ALU.

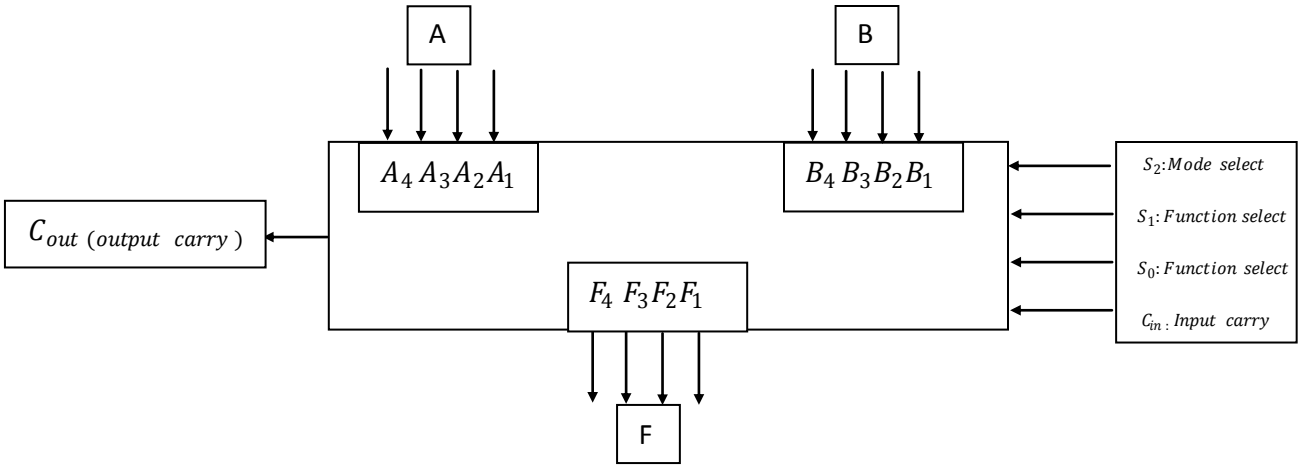
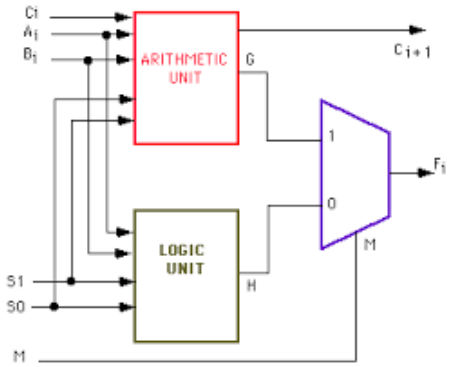


Figure: Block diagram of 4 bit ALU

S1	S0	Arithmetic (s2=0)	Logical(s2=1)
0	0	Addition	OR
0	1	Subtraction	AND
1	0	Multiplication	NOT
1	1	Division	Ex-OR

The selection lines are decodes within the ALU so that the K selection variables can specify up to 2^k distinct operations. Figure shows 4 bit inputs A and B to generate 4 bit result F. The mode select distinguish between arithmetic and logic operations. The two function select inputs specify the particular arithmetic or logical operations.



3.5 Design principles of Modern System

- Efficient memory utilization.
- Use of caches, different levels.
- Improvement in input and output mechanism.
- Pipelining.
- Parallel processing.
- Multi-core systems.