Types of CPU Organizations

- Single Accumulator based CPU
- Stack Based CPU
- General Register Based CPU

Accumulator Based CPU

- The computers, present in the early days of computer history, had accumulator based CPUs.
- In this type of CPU organization, the accumulator register is used implicitly for processing all instructions of a program and store the results into the accumulator.
- The instruction format that is used by this CPU Organisation is **One address field**. Due to this the CPU is known as **One Address Machine**.

The main points about Single Accumulator based CPU Organisation are:

- In this CPU Organization, the first ALU operand is always stored into the Accumulator and the second operand is present either in Registers or in the Memory.
- Accumulator is the default address thus after data manipulation the results are stored into the accumulator.
- One address instruction is used in this type of organization.

Mainly two types of operation are performed in single accumulator based CPU organization:

- Data transfer operation –
- In this type of operation, the data is transferred from a source to a destination.
- For ex: LOAD X, STORE Y
- Here LOAD is memory read operation that is data is transfer from memory to accumulator and STORE is memory write operation that is data is transfer from accumulator to memory.

- ALU operation –
- In this type of operation, arithmetic operations are performed on the data.
- For ex: MULT X
- where X is the address of the operand. The MULT instruction in this example performs the operation,

- AC <-- AC * M[X]
- AC is the Accumulator and M[X] is the memory word located at location X.

- This type of CPU organization is first used in PDP-8 processor and is used for process control and laboratory applications. It has been totally replaced by the introduction of the new general register based CPU.

- **Advantages –**
- One of the operands is always held by the accumulator register. This results in short instructions and less memory space.
- Instruction cycle takes less time because it saves time in instruction fetching from memory.
- **Disadvantages –**
- When complex expressions are computed, program size increases due to the usage of many short instructions to execute it. Thus memory size increases.
- As the number of instructions increases for a program, the execution time increases.

Stack Organisation

- The computers which use Stack-based CPU Organization are based on a data structure called **stack**
- The stack is a list of data words. It uses **Last In First Out (LIFO)** access method which is the most popular access method in most of the CPU.
- A register is used to store the address of the topmost element of the stack which is known as **Stack pointer (SP)**. In this organisation, ALU operations are performed on stack data.
- It means both the operands are always required on the stack. After manipulation, the result is placed in the stack. (Zero Address)

The main two operations that are performed on the operators of the stack are **Push** and **Pop**.

- Push –
- This operation results in inserting one operand at the top of the stack and it decrease the stack pointer register. The format of the PUSH instruction is:
- PUSH
- It inserts the data word at specified address to the top of the stack. It can be implemented as:

- //decrement SP by 1
- SP <-- SP - 1

- //store the content of specified memory address
- //into SP; i.e, at top of stack
- SP <-- (memory address)

- Pop –
- This operation results in deleting one operand from the top of the stack and it increase the stack pointer register. The format of the POP instruction is:
- POP
- It moves the data word at the top of the stack to the specified address. It can be implemented as:

- //transfer the content of SP (i.e, at top most data)
- //into specified memory location
- (memory address) <-- SP

- //increment SP by 1
- SP <-- SP + 1

- Operation type instruction does not need the address field in this CPU organization. This is because the operation is performed on the two operands that are on the top of the stack. For example:

- SUB
- This instruction contains the opcode only with no address field. It pops the two top data from the stack, subtracting the data, and pushing the result into the stack at the top.

- PDP-11, Intel’s 8085 and HP 3000 are some of the examples of the stack organized computers.

- The advantages of Stack based CPU organization –

- Efficient computation of complex arithmetic expressions.
- Execution of instructions is fast because operand data are stored in consecutive memory locations.
- Length of instruction is short as they do not have address field.
- The disadvantages of Stack based CPU organization –

- The size of the program increases.

General Register based CPU Organization

- When we are using multiple general purpose registers, instead of single accumulator register, in the CPU Organization then this type of organization is known as General register based CPU Organization.
- In this type of organization, computer uses two or three address fields in their instruction format. Each address field may specify a general register or a memory word.
- If many CPU registers are available for heavily used variables and intermediate results, we can avoid memory references much of the time, thus vastly increasing program execution speed, and reducing program size.

- For example:
- MULT R1, R2, R3

- R1 <-- R2 * R3
- This instruction also can be written using only two address fields as:

- MULT R1, R2
- In this instruction, the destination register is the same as one of the source registers. This means the operation

- R1 <-- R1 * R2
- The use of large number of registers results in short program with limited instructions.

- Some examples of General register based CPU Organization are IBM 360 and PDP- 11.

- The advantages of General register based CPU organization –

- Efficiency of CPU increases as there are large number of registers are used in this organization.
- Less memory space is used to store the program since the instructions are written in compact way.

- The disadvantages of General register based CPU organization –

- Care should be taken to avoid unnecessary usage of registers. Thus, compilers need to be more intelligent in this aspect.
- Since large number of registers are used, thus extra cost is required in this organization.

- General register CPU organisation is of two type:

- Register-memory reference architecture (CPU with less register)– In this organisation Source 1 is always required in register, source 2 can be present either in register or in memory. Here two address instruction format is the compatible instruction format.
- Register-register reference architecture(CPU with more register)– In this organisation ALU operations are performed only on a register data. So operands are required in the register. After manipulation result is also placed in register. Here three address instruction format is the compatible instruction format.

X = (A + B) * (C + D).

- Complete the operation using
- 3 address instruction
- 2 address instruction
- 1 address instruction

- ADD R1, A, B R1 ← M [A] + M [B]
- ADD R2, C, D R2 ← M [C] + M [D]
- MUL X, R1, R2 M [X] ← R1 ∗ R2

- MOV R1, A R1 ← M [A]
- ADD R1, B R1 ← R1 + M [B]
- MOV R2, C R2 ← M [C]
- ADD R2, D R2 ← R2 + M [D]
- MUL R1, R2 R1 ← R1∗R2
- MOV X, R1 M [X] ← R1

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

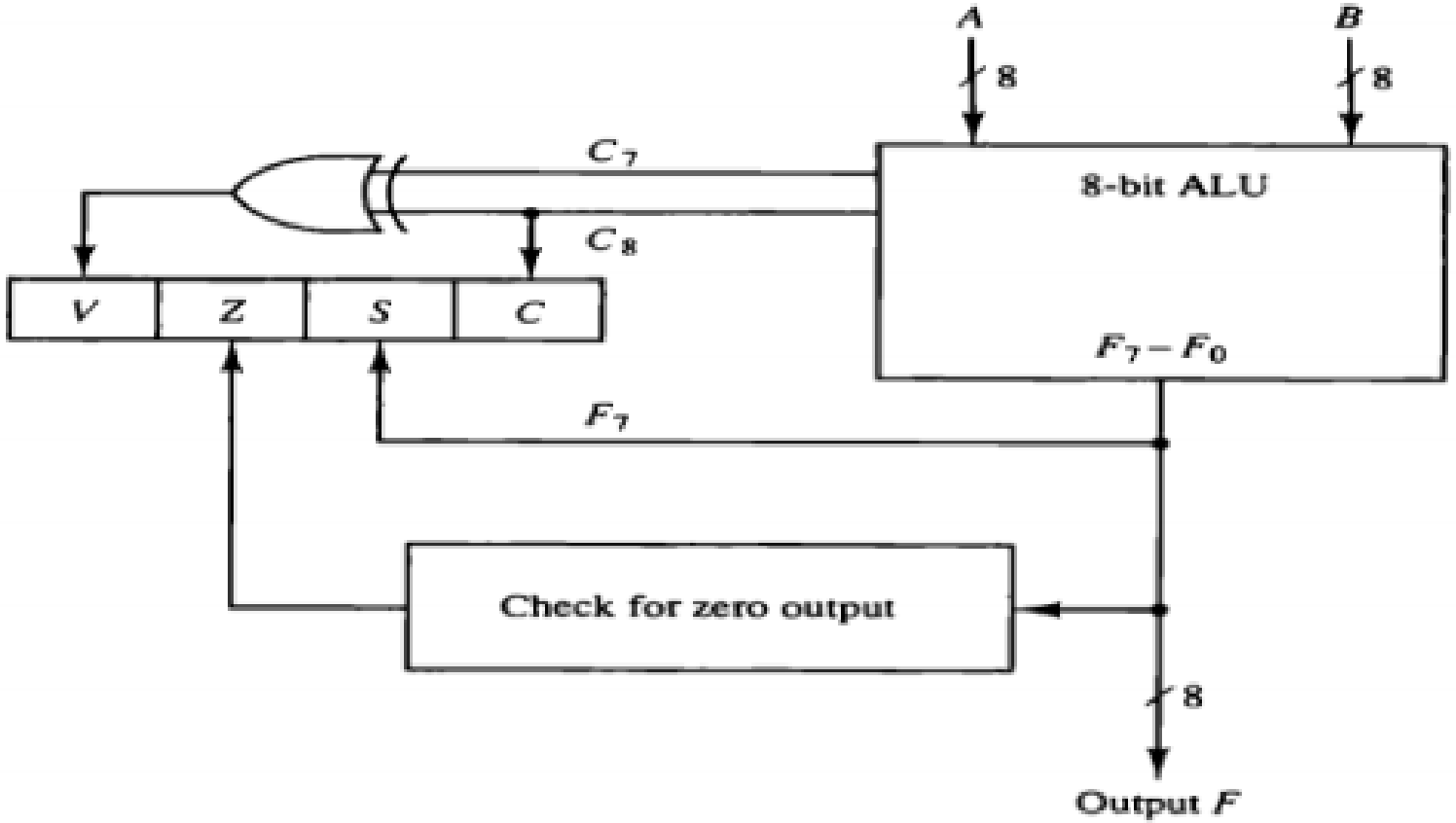
- PUSH A TOS ← A
- PUSH B TOS ← B
- ADD TOS ← (A + B)
- PUSH C TOS ← C
- PUSH D TOS ← D
- ADD TOS ← (C + D)
- MUL TOS ← (C + D) ∗ (A + B)
- POP X M [X] ← TOS

Status Bit Conditions

- It is sometimes convenient to supplement the ALU circuit in the CPU with a status register where status bit conditions can be stored for further analysis. Status bits are also called condition-code bits or flag bits. The four status bits are symbolized by C, S, Z, and V. The bits are set or cleared as a result of an operation performed in the ALU.

- 1. Bit C (carry) is set to 1 if the end carry C8 is 1. It is cleared to 0 if the carry is not generated.
- 2. Bit S (sign) is set to 1 if the MSB is 1. It is set to 0 if the bit is 0.
- 3. Bit Z (zero) is set to 1 if the output of the ALU contains all 0's. It is cleared to 0 otherwise. In other words, Z = 1 if the output is zero and Z = 0 if the output is not zero

- 4. Bit V (overflow) is set to 1 if the exclusive-OR of the last two carries is equal to 1, and Cleared to 0 otherwise. This is the condition for an overflow when negative numbers are In 2's complement. For the 8-bit ALU, V = 1 if the output is greater than +127 or less than -128.



Status register bits.