DAY 17-111 DAYS VERIFICATION CHALLENGE

Topic : Instructions & Buses

Skill : Computer Architecture

DAY 17 CHALLENGE:

1. Can you tell at least 5 common rules of assembly language?

- **Instruction Format**: Follow the specific format for instructions, including mnemonics and operands.
- **Registers and Memory Usage**: Correctly use and manage registers and memory.
- **Labelling and Control Flow**: Use unique labels for jumps and loops to manage control flow.
- **Comments and Documentation**: Add comments to improve readability and maintenance.
- **Directives**: Use assembler directives (e.g., ORG, DB, EQU) to manage data and control the assembly process.

2. Explain 3 major fields in an instruction the operation field, the address field, and the mode field.

- **Operation Field (Opcode)**: This field specifies the operation to be performed by the instruction. It tells the CPU what action to take (e.g., add, move, load, store, jump). The operation field is usually represented by a mnemonic, such as MOV, ADD, SUB, JMP.
- **Address Field**: The address field specifies the location of the data required for the operation. This can be a direct address, a register, or an indirect address. The address field tells the CPU where to find the operands needed for the operation. For example, in MOV AX, [1234], 1234 is the address where the data is located.
- **Mode Field**: The mode field (or addressing mode) specifies how the address field should be interpreted. It defines how the operand is accessed (e.g., direct, indirect, immediate, register, indexed). For example, immediate mode uses the value directly in the instruction (MOV AX, 5), while register mode uses a value from a specified register (MOV AX, BX).

3. Explain types of micro-operations (Register transfer, Shift, Logic, Arithmetic) with example?

- **Register Transfer Micro-operations**:
  - These operations involve transferring data from one register to another.
  - Example: R2 <- R1 means the contents of register R1 are transferred to register R2.
- **Shift Micro-operations**:
  - These operations involve shifting the bits of a register left or right.
  - Example: R1 <- shl R1 means the contents of register R1 are shifted left by one bit.
    - If R1 initially contains 1010, after the shift, R1 will contain 0100.
- **Logic Micro-operations**:
  - These operations perform bitwise logical operations on the data in registers.
  - Example: R1 <- R2 AND R3 means the contents of register R2 are logically ANDed with the contents of register R3, and the result is stored in register R1.
    - If R2 contains 1010 and R3 contains 1100, then R1 will contain 1000.
- **Arithmetic Micro-operations**:
  - These operations perform arithmetic operations on the data in registers.
  - Example: R1 <- R2 + R3 means the contents of register R2 are added to the contents of register R3, and the result is stored in register R1.
    - If R2 contains 5 and R3 contains 3, then R1 will contain 8.

4. What is a horizontal microcode? Why do we use it?

Horizontal microcode is a method used in microprogrammed control units to define the control signals necessary to execute each micro-operation within the CPU.

Horizontal Microcode

1. **Definition**:

    - Horizontal microcode consists of wide microinstructions where each bit represents a distinct control signal. This type of microcoding uses a long instruction word, often containing several bits, each corresponding to a specific control signal or set of signals in the CPU.
2. **Structure**:
    - Each microinstruction directly specifies which control lines are activated for a given micro-operation. Because each bit controls a specific line, the width of the microinstruction is typically large.
3. **Example**:
    - Suppose a microinstruction is 32 bits wide, where each bit controls a specific part of the CPU, such as enabling registers, setting ALU operations, or controlling buses.

Why Use Horizontal Microcode?

1. **Parallelism**:
    - Horizontal microcode allows for high levels of parallelism since multiple control signals can be activated simultaneously. This enables the execution of multiple micro-operations at the same time, which can improve the performance of the CPU.
2. **Flexibility and Control**:
    - It provides fine-grained control over the CPU's internal operations, making it possible to optimize the micro-operations for specific tasks or hardware configurations.
3. **Efficiency**:
    - By directly specifying control signals, horizontal microcode can lead to more efficient hardware utilization, as unnecessary operations can be avoided, and specific control paths can be optimized.

5. Explain 3 major types of buses - address, data & control bus with an example.

1. Address Bus

Definition:

- The address bus carries the addresses of memory locations where data is to be read from or written to.

Function:

- It specifies the source or destination address in memory.

Example:

- If a CPU wants to read data from memory location 0x1A3F, it sends the address 0x1A3F over the address bus to the memory controller. The width of the address bus determines the maximum amount of memory that can be addressed. For example, a 32-bit address bus can address 2^32 (4,294,967,296) unique memory locations.

2. Data Bus

Definition:

- The data bus transfers actual data between the CPU, memory, and peripheral devices.

Function:

- It carries the data that is being processed or transferred.

Example:

- After the address bus specifies the memory location 0x1A3F, the data stored at that location, say 0x5A, is transferred to the CPU via the data bus. If the data bus is 8 bits wide, it can transfer one byte of data at a time; if it is 32 bits wide, it can transfer four bytes at a time.

3. Control Bus

Definition:

- The control bus carries control signals that coordinate and manage the various operations of the CPU and other components.

Function:

- It transmits commands, timing, and specific control signals to ensure proper data transfer and operation.

Example:

- When the CPU reads data from memory, it sends a read command signal over the control bus. Conversely, when writing data to memory, it sends a write command signal. Other control signals might include interrupt requests, clock signals, and status signals.

6. Give one example of below instructions: -

a. Arithmetic Instructions.

ADD, SUB, MUL, ADC, SUBB…

b. Branch Instructions.

BRNE, BREQ, JMP…

c. Data Transfer InsStructions.

MOV, LDA, STR…

d. Logic Instructions.

AND, OR, XOR, NOT…

e. bit oriented instruction

CBI, SBI, SBIS, CBIS…