**Assignment -3**

**Student Name:** Sagar Saini **UID:** 21BCS10741

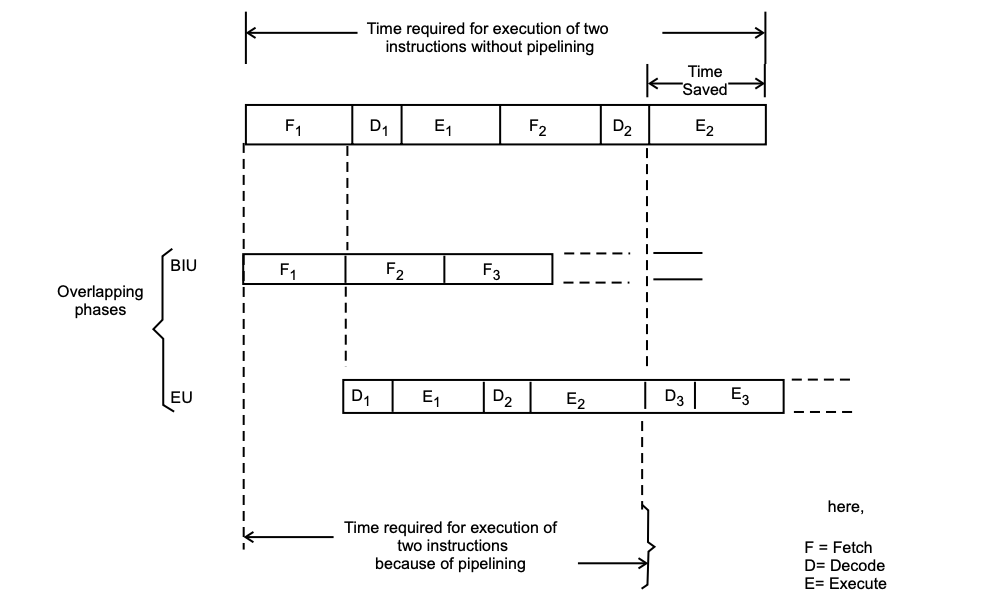
**Branch:** Computer Science.  **Section/Group:** 809A

**Semester:** 4th **Date of Performance:** 5-May-22

**Subject Name:** MPI Lab **Subject Code:** 20CSP-253

**Q1.** **Analyse the working of instruction queue in 8086?**

**Answer:** The instruction queue is 6-bytes in length, operates on FIFO basis, and receives the instruction codes from memory. BIU fetches the instructions meant for the queue ahead of time from memory. In case of JUMP and CALL instructions, the queue is dumped and newly formed from the new address. Because of the instruction queue, there is an overlap between the instruction execution and instruction fetching. This feature of fetching the next instruction when the current instruction is being executed, is called Pipelining.



Which is self-explanatory, shows that there is definitely a time saved in case of overlapping phases (as in the case of 8086) compared to sequential phases (as in the case of 8085). Initially, the queue is empty and CS : IP is loaded with the required address (from which the execution is to be started). Microprocessor 8086 starts operation by fetching 1 (or 2) byte(s) of instruction code(s) if CS : IP address is odd (even). The 1st byte is always an opcode, which when decoded, one byte in the queue becomes empty and the queue is updated. The filling in operation of the queue is not started until two bytes of the instruction queue is empty. The instruction execution cycle is never broken for fetch operation. After decoding of the 1st byte, the decoder circuit gets to know whether the instruction is of single or double opcode byte. For a single opcode byte, the next bytes are treated as data bytes depending upon the decoded instruction length, otherwise the next byte is treated as the second byte of the instruction opcode.

**Diagram, schematic

Description automatically generated**

For a 2-byte instruction code, the decoding process takes place taking both the bytes into consideration which then decides on the decoded instruction length and the number of subsequent bytes which will be treated as instruction data. Updation of the queue takes place once a byte is read from the queue.

**Q2 Discuss the functions of all general purpose registers of 8086.Explain the special function of each register and instruction support for these functions.**

**Answer:** General Purpose registers are used for temporary storage of data and memory access. Since the processor accesses register more quickly than memory. 8086 has four 16-bit general-purpose registers AX, BX, CX and DX. These are available to the programmer, for storing values during programs. Each of these can be divided into two 8-bit registers such as AH, AL; BH, BL; CL, CH and DL, DH. Beside their general use, these registers also have some specific functions.

* **AX Register (16-Bits):** It holds operands and results during multiplication and division operations. All IO data transfers using IN and OUT instructions use A register (AL/AH or AX). It functions as accumulator during string operations.

**Example:**

MUL BL ; AX = (AL × BL)

MUL BX ; DX-AX = (AX × BX)

MUL BYTE PTR [BX] ; AX = (AL x DS:[BX])

* **BX Register (16-Bits):** It holds the memory address (offset address), in Indirect Addressing modes.

**Example:**

MOV CL, [BX] ; Moves a byte from the address pointed by BX in Data Segment into CL. Physical Address calculated as DS \* 10H + BX

MOV CH, [BX+6] ; Moves a byte from the address pointed by BX+6 in Data Segment to CH; Physical Address: DS \* 10H + BX + 6H

MOV CL, [BX+SI] ; Moves a byte from the address pointed by BX+SI in Data Segment to CL. Physical Address: DS \* 10H + BX + SI

* **CX Register (16-Bits):** It holds count for instructions like: Loop, Rotate, Shift and String Operations.

**Example:**

MOV CX, 40H

BACK: MOV AL, BL

ADD AL, BL

. . MOV BL, AL

LOOP BACK

MOV CX, 40H BACK: MOV AL, BL ADD AL, BL . . MOV BL, AL LOOPZ BACK

* **DX Register (16-Bits):** It is used with AX to hold 32 bit values during Multiplication and Division. It is used to hold the address of the IO Port in indirect IO addressing mode.

**Example:**

¬ MUL BX ; DX-AX = (AX × BX)

¬ MOV DX, 2000H

IN AL, DX

**Learning outcomes (What I have learnt):**

1. Learned about the 8086 processor .
2. Learned about the 8086 architecture.
3. Learned about the pin diagram of 8086.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |