**ASSEMBLY 1 PROJECT**

**CS-550**

**Name:-Janki J Patel**

**CWID : 10457365**

**Professor : Ed Banduk**

**1). What is the 68000 processor?**

Motorola 68000 is a processor is a 16/32 bit CISC processor planned by Motorola and dispatched in the year 1979. Inside the processor has 32 bit information and address transport. Remotely the processor has a 16 bit information transport and a 24 cycle address transport. This restricts the size of principle memory to 16MB. It has a speed of up to 20 MHz.

Fig 1. Motorola 6800 processor. Source Wikipedia.[2]

Motorola 68000 was utilized in an assortment of PCs prominently Apple Lisa 2, Apple Macintosh 128, Atari 520STfm and so forth. Following 30 years the 68000 processor is as yet produced by different makers over the globe.[2][5]

Addressing modes in Motorola 68000 processor

Where should we get our data from and where our commands should store our data.

Natural tending to modes: In characteristic tending to mode the operand area is suggested by the operation code of the guidance. For e.g.: CLA could mean Clear Accumulator. Here the location of the aggregator need not be indicated.[4]

Immediate addressing mode**:** Prompt tending to is the place a fixed worth is utilized as a source.

For e.g.: MOVE.L #$A56C20, D3

Stores the Hexadecimal number in D3 register

Result D3 = $A56C20

Relative Addressing mode: In relative tending to mode locations can be stacked anyplace in the memory without modifying the addresses. Relative tending to is position autonomous. Another favorable position is that length of guidance will be shorter.

For e.g.: MOVE.L COUNT(PC), D0. Stores estimation of Program Counter in Register D0. [4]

Extended addressing mode: Broadened tending to mode is utilized when the location of the memory area is too huge to even consider being put away inside 1 byte(8 digit). Two bytes can be deciphered as on address to store tends to more prominent than scope of 0-255.[4]

Indexed addressing mode: This tending to mode is utilized in exhibits. The base location is added to a consistent which is the following location.

For e.g.: if a variety of numbers is to be put away and the base location is 1000. Since the size of a whole number is 4 bytes the following location is 1004. Along these lines address for cluster components are resolved. [4]

**2).What is 68000 assembly language?**

Fundamental activities of 68000 low level computing constructs incorporate.

Information Movement guidelines: For e.g.: MOVE #5,D1 Register D1 is stacked with number 5.

Examination directions: For e.g.: CMP #5, D1 contrast the number 5 and substance of D1 Register

Program control directions: BNE EXIT\_IF, If not equivalent at that point go to mark EXIT\_IF

Whole number Arithmetic guidelines: For e.g.: ADD D3,D7 Adds information present in both D3 and D7 and stores the outcome in D7.[1]

**3).Why we are using Assemblers?**

A constructing agent is a PC program that deciphers the code written in low level computing construct into machine language. Directions just written in machine language can be executed by PCs. We use constructing agents since a developer can't program in machine language which just comprises of 1's and 0's. Low level computing construct is changed over into Object code by making an interpretation of guidelines into mathematical proportionate. Portrayal incorporates activity code and other information. [6]

**4).What is 68000 Simulator?**

A test system is a PC program which mirrors the genuine framework. A 68000 test system helps in imagining the condition of the CPU, substance of memory and registers. A test system encourages the developer to troubleshoot the program before the program is run on the chip. Call stack, neighborhood variable, worldwide factors can likewise be seen by the software engineer when the program is being run. A software engineer can reenact exemptions ie which are the conditions when chip will close down if special cases are not taken care of appropriately. Since 68000 is a chip generally utilized across numerous gadgets a test system will help in distinguishing programming mistakes all the more proficiently. There is no compelling reason to run a program on a chip more than once in the event that we have a test system with us. This spares time as we don't need to stack our program on the chip each time it changes. [7]

References

[1] Skinner, T. P. (1988). *Assembly language programming for the 68000 family*. New York: John Wiley &       Sons.

[2] Motorola 68000. Retrieved from <https://en.wikipedia.org/wiki/Motorola_68000>.

[3]Savadjiev, P. Retrieved from <https://www.cs.mcgill.ca/~cs573/fall2002/notes/lec273/lecture9/#bins>.

[4] Integrated Publishing, Inc. - A (SDVOSB) Service-Disabled Veteran Owned Small Business. Retrieved from <http://firecontrolman.tpub.com/14100/css/Extended-Operand-Addressing-215.htm>.

[5] CPU World. Retrieved from <http://www.cpu-world.com/CPUs/68000/index.html>

[6] Assembly language. Retrieved from <https://en.wikipedia.org/wiki/Assembly_language#Assembler>.

[7]Crossware.  Retrieved from <https://www.crossware.com/m68xxx/Simulator>