Stevens Institute of Technology

Assembly Project 1

Introduction to

68000 processor

Jay Kalyanbhai Savani

Cwid - 20009207

1. **What is the 68000 Processor? (mentioning about 68000 addressing modes: inherent, immediate, relative, extended, indexed)**

   The first 16-bit and 32-bit microprocessors were introduced with the 68000 processor. It only has a 16-bit arithmetic processing unit and an external data bus despite being a 32-bit register. It lacks the bad segment registers of Intel's modern CPUs, which make programming them difficult. It has a linear space with a 24-bit addressing width. So, a single directly accessible array or structure can be bigger than 64KB. In order to fit the address bus into a 64-pin package, the top 8 bits of addresses that are calculated as 32-bits are trimmed. Eight data registers, designated as D0-D7, are present in the 68000 processor and are used to store numbers that will undergo a variety of logical and mathematical operations. Seven address registers are also present: the normal range of A0-A6 is used as pointers. One stack pointer is currently in use: SP, often known as A7. addressing styles:

   1. **Register direct addressing**:
      a. **Data register direct**: The operand is indicated by the letter "Dn" and is located in the data register. The actual address is En=Dn.
      b. **Address register direct**: In this case, the operand is indicated by the letter "An" and is located in the address register. The actual address is En=An.
   2. **There are two types of index addressing modes**:
      a. **Basic index mode**: The effective address is created in the basic index mode by adding a constant value to the content (which must be 16-bits) of an address register.

b. **Full index mode**: In this mode, the effective address is created by adding a constant value to the content of a 2-address register—the content must be 8 bits long.

3. **Inherent**: Most instructions in this mode are only one bit long and do not require the retrieval of any operands or an operand address. For instance, CLRA: Clear the A register
4. **Immediate Addressing Mode**: The operand for immediate instructions is contained in the bytes after the opcode, this mode has a constant operand. For instance: LDJ #95: Load accumulator with 95.
5. **Extended addressing mode**: All instructions in this mode are 3-byte long and can place their operand at any address within a 64-kilobyte memory map. LDJ $100, for instance, means to load J with the items at location 100.

2. What is the 68000 Assembly Language?

68000 assembly is the assembly language used for the Motorola 68000, or commonly known as the 68K. It should not be confused with the 6800 (which predates it). The Motorola 68000 is a big-endian processor with full 32-bit capabilities (despite most systems that use it being considered 16-bit.) It was used in many computers such as the Amiga or the Canon Cat, as well as game consoles such as the Sega Genesis and Neo Geo. The 68000 has several different versions, which include the 68008, 68010, and 68012. The 68000 and 68010 are packaged either in a 64-pin DIP (dual in-line package) with all pins assigned or in a 68-pin quad pack or PGA (pin grid array) with some unused

pins. The 68000 is also packaged in 68-terminal chip carrier. The 68008 is packed in a 48-pin dual in-line package, whereas the 68012 is packed in an 84-pin grid array. The 68008 provides the basic 68000 capabilities with inexpensive packaging. It has an 8-bit data bus, which facilitates the interfacing of this chip to inexpensive 8-bit peripheral chips.

3. Why we are using Assemblers?

An assembly language, also known as an assembler language, is a low-level programming language. Assembly language has a very strong correspondence with the architecture's machine code instruction and is specific only to that machine. Therefore, different machines have different assembly languages. This type of language makes use of symbols to represent an operation or instruction. Hence, it is also often known as symbolic machine code. Despite the prevalence of high-level languages that are mainly used for the development of applications and software programs, the importance of assembly language in today's world cannot be understated. A programmer can still gain a lot if he/she can learn to code in assembly language and implement it. These days, assembly language makes it possible to manipulate hardware directly, address critical issues concerning performance and also provide access to special instructions for processors. Uses of assembly language include coding device drivers, real-time systems, low-level embedded systems, boot codes, reverse engineering and more. When compared to high-level languages, which are mostly in the form of abstract data types,

assembly language is bare and transparent. This is largely since it has a small number of operations. So, this is very helpful for algorithm analysis, consisting of semantics and flow of control. It also makes it easier for debugging, as it is less complex.

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

This 68000 provides a virtual 68000 microprocessor that runs the computer or PC, allowing the programmer to run the 68000 program without any 68000 hardware and execute in Windows graphics or in other environments or in different operating systems without encountering any errors. It is a tool that enables the computer programmer to test and create assembly code. In Windows, for instance, Cross ware S68kNT can be used to simulate a 68000 microprocessor. By utilizing this stimulator, users can access a variety of new features, such as seamless integration with the embedded programming Studio, source code profiling, full source level debugging, and more.

REFERENCES

S. A. Mengel and J. M. Conrad, "Motorola 68000 family simulators in education," Proceedings of 1994 IEEE Frontiers in Education Conference - FIE '94, 1994, pp. 106-110, doi: 10.1109/FIE.1994.580480.

Antonakos, J. L. (2004). *The 68000 microprocessor: Hardware and software principles and applications*. Pearson Prentice Hall.

Leventhal, L. A. (1986). *68000 Assembly language programming*. McGraw-Hill.