Assembly 1

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

the 68000 Processor is a 16/32-bit CISC microprocessor, which implements a 32-bit instruction set, with 32-bit registers and 32-bits internal data bus, but with a 16-bit data ALU and two 16-bit arithmetic ALUs and a 16-bit external data bus, designed and marketed by Motorola Semiconductor Products Sector.

A key concept in computing in both high-languages and low-level languages is the addressing mode. Computers perform operations on data and you have to specify where the data comes from. The various ways of specifying the source or destination of an operand are called addressing modes. And 68000 has these basic address modes.

1. Absolute addressing (the operand specifies the location of the data): In absolute addressing you specify an operand by providing its location in memory or in a register.

2. Immediate addressing (the operand provides the operand itself): In immediate addressing the operand is an actual value rather than a reference to a memory location. The 68K assembler indicates immediate addressing by prefixing the operand with the '#' symbol.

3. Indirect addressing (the operand provides a pointer to the location of the data): Indirect addressing specifies a *pointer* to the actual operand which is invariably in a register. For example, the instruction, MOVE (A0),**D1** , first reads the contents of register A0 to obtain a pointer that gives you the address of the operand. Then it reads the memory location specified by the pointer in A0 to get the actual data.

4. Inherent addressing mode: Its effective address is that in processor register (CCR, SR, SP, USP, SSP or PC) or memory but not indicated in the instruction. Furthermore, it format is that usually no operand. For example, RTS.

5. Relative addressing mode: Its effective address is that calculated by adding a displacement to the PC. This is particularly useful in connection with jumps, because typical jumps are to nearby instructions (in a high-level language
   most **if** or **while** statements are short). Measurements of actual programs suggest that an 8- or 10-bit offset is large enough for some 90% of conditional jumps (±128 or ±512 bytes).

6. Extended addressing mode: It uses a 16-bit address. And size of data depends on the op code. The addressing mode EXT is identified and distinguished from the other addressing

modes by the presence of a 16-bit address of the 2nd operand, and the absence of the # symbol to distinguish it from the IMM mode.

7. Index addressing mode: The address of the 2nd operand is also called the “effective address.” The effective address is computed (generated) by value of the index register X + the unsigned 16-bit extension of the offset. The address of the 2nd operand is a variable, since the value of index register X can be changed, through several different instructions.

2. What is the 68000 Assembly Language?

The assembly language is the assembler processes the source program line by line. A line of the source program can be translated into a machine instruction or generate an element or elements of data to be placed in memory; or generate an element or elements of data to be placed in memory; or the line may only provide information to the assembler itself. The line of the source program is sometimes referred to as source statements.

Regardless of the use of a particular line of the source program, the format of each line is standard. The general format of a source line consists of four fields, as follows:

[<label>] <operation [<operand>] [<comment>]

3. Why we are using Assemblers?

Most programs use addresses within the program as data storage or as targets for jumps or calls. When programming in machine language, these addresses must be calculated by hand. The assembler solves this problem by allowing the programmer to assign a symbol to an address. The programmer may then reference that address elsewhere by using the symbol. The assembler computes the actual address for the programmer and fills it in automatically.

To handle numbers, most assemblers, including the 68000, consider all numbers as decimal numbers unless specified otherwise.

4. What is the 68000 Simulator?

The 68000 simulator provides two devices that can be attached to the microprocessor. These devices are the M68681 Dual UART and RAM. Each of these devices is described below.

M68681 Dual UART Device: The M68681 Dual Universal Asynchronous Receiver/Transmitter (DUART) device is modeled after the Motorola MC68681 DUART.

RAM Device: The RAM device allows you to attach a specified amount of memory to the 68000. When the RAM device is attached to the simulator, using the Edit Setup option. RAM device allows you to enter the base address and size of the RAM module. Both of these values are in hexadecimal.

Reference:

1. Starnes, Thomas W. (April 1983). “Design Philosophy Behind Motorola’s MC68000” Vol. 8 no. 4, Retrieved 6/19/2018
2. Muhammad Mun’im Ahmad Zabidi SEE 3223 Microprocessor Systems 2: 68000 Architecture http://ocw.utm.my/pluginfile.php/1305/mod_resource/content/0/02-68k-Architecture.ppt.pdf *accessed 28 September 2018*
3. Alan Clements 68k Addressing Modes http://alanclements.org/68kaddressingmodes3.html *accessed 28 September 2018*
4. The – Thomas P. Skinner *Assembly Language Programming for the 68000 Family*, Page 29
5. The – Thomas P. Skinner *Assembly Language Programming for the 68000 Family*, Page 71
6. Bradford W. Mott Chapter 3 Motorola 68000 Simulator and Assembler https://www.nada.kth.se/hacks/doc/bsvc/chap3.htm *accessed 28 September 2018*