68000 PROCESSOR

Thanapoom Phatthanaphan

Department of Computer Science, Stevens Institute of Technology

CS 550: Computer Organization and Programming

Dr. Edward Banduk

November 5, 2022

The 68000 Processor

The 68000 processor is Motorola's first 16-bit microprocessor. Its address and data registers are all 32 bits wide, and its ALU is 16 bits wide. The 680000 requires a single 5-V supply. The processor can be operated from a maximum internal clock frequency of 26 MHz. The 68000 is available in several frequencies, including 4, 6, 8, 10, 12.5, 16.67, and 25 MHz. The 68000 does not have on-chip clock circuitry and therefore, requires an external crystal oscillator or clock generator/driver circuit. The 68000 is a general-purpose register-based microprocessor. Although the 68000 PC is 32 bits wide, only the low-order 24 bits are used. Because this is a byte-addressable machine, it follows that the 68000 microprocessor can directly address 16 MB of memory. Note that brackets [1, are used in the examples throughout this chapter to indicate the contents of a 68000 register or a memory location. There are many different addressing modes in the 68000 processor. These are 5 examples of addressing modes in the 68000 processor.

1. Inherent addressing mode

The locations of data are implied by the instruction name. For example, ABA, which adds register B to A, implied the registers in the instruction name. There is no operands needed to indicate memory to work with or constant value to load.

2. Immediate addressing mode

The actual data is located in memory immediately following the instruction's opcode. The instruction name is followed by a constant value or label with a # sign in front.

3. Relative addressing mode

The value in the operand is computed from the relationship between where you are currently in the code in memory to where you are going to, the label. the instruction name followed by a 8 bit value that indicates the distance to move (add to PC) from where

currently in code. The relative address value is in two's complement form so it can only indicate a change of -128 memory locations backwards and +127 memory locations forwards.

4. Extended addressing mode

The address of the data is located in memory immediately following the instruction's opcode. the instruction name is followed by a label that is in the assembly program somewhere, or is followed by a constant value, without a # sign.

5. Indexed addressing mode

The value in an index register (X or Y) is added to an offset value, which provides the final effective address of the data. the instruction name followed by a value, register pair.

The 68000 Assembly Language

Microprocessors are typically programmed using semi-English-language statements (assembly language). Programs in assembly is represented by instructions that use English-language-type statements. The programmer finds it relatively more convenient to write programs in assembly than in machine language. The assembly language program is translated into binary via a program called an assembler. The assembler program reads each assembly instruction of a program as ASCII characters and translates them into the respective binary op-codes. For example, the 68000 assembler translates the RTS (Return from subroutine) instruction into its 16-bit binary op-code is 0100111001110101 (4E75 in hex) as shown in picture below.

Conversion of RTS into Its Binary Op-Code

Assembly Code	Binary Form of ASCII Codes as Seen by the Assembler	Binary Op-Code Created by the 68000 Assembler
R T S	0101 0010 0101 0100 0101 0011	0100 1110 0111 0101

Why are we using Assemblers?

An advantage of the assembler is address computation. 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. Therefore, with assembly language, the programmer does not have to find the numerical op-codes from a table of the instruction set, and programming efficiency is improved significantly.

The 68000 Simulator

Developing 68000 assembly language programs, the 68000 assembler and simulator (debugger) are used. The assembler converts the source file (.ASM file) into an object file (.OBJ file) containing the binary codes or machine codes that the 680000 will understand.

BIBLIOGRAPHY

Rafiquzzaman, M. (2008). *Microprocessor Theory and Applications with 68000/68020 and Pentium.* John Wiley & Sons, Inc.

Miller, H. G. (1999). *Microcomputer Engineering* (2nd ed.). Prentice-Hall, Inc.