LAB 1: MC68000 CPU PROGRAMMING
DATA TRANSFER INSTRUCTIONS

OBJECTIVES

At the end of the laboratory works, you should be able to write simple assembly language programs for the MC68000 CPU using data transfer instructions and execute the programs on the MC68000 trainer.

INTRODUCTION

The Motorola MC68000 Central Processing Unit has 16-bit data bus and a 24-bit address bus which is capable of accessing a linear address space of 16 Mbytes. The MC68000 Trainer Board consists of two types of memory; ROM and RAM. ROM consists of monitor program and starts at location 000000H-07FFFFH. The monitor program is the program that controls the resources available on the Trainer Board. User programs are written into the RAM. This memory starts at address 400400H up to 43FFFFH. Therefore, user programs should be started at address 400400H.

In order to write assembly language programs for the MC68000 CPU, it is important to understand the programming model of the CPU, or also known as the register set.

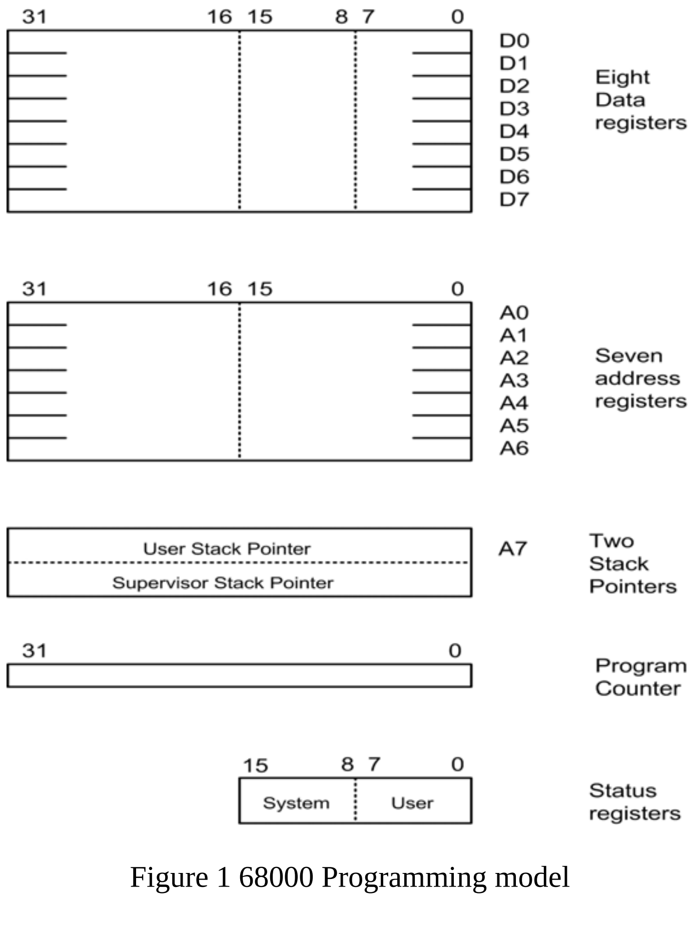


Figure 1 68000 Programming model

The MC68000 CPU have eight data registers, D0 to D7, eight address registers, A0 to A7; a program counter (PC); and status register (SR) as shown in Figure 1. All registers except for the status register are 32 bit in length. The data registers are used to store information (data) within the MC68000 CPU itself. Address registers are used to store the location (memory address) where data can be found outside the CPU, in the external memory chips.

There are four types of data transfer instructions provided by 68000 microprocessor:
a) register to register data transfer
b) memory to register data transfer
c) register to memory data transfer
d) memory to memory data transfer

Data transfer instructions involve instructions such as MOVE, MOVEA, MOVEM, MOVEQ, SWAP etc to copy or move data between the various locations.

EQUIPMENTS

1. A Personal computer installed with the MC68000 CPU Editor/Assembler and Simulator software, EASy68K.
2. MC68000 CPU Trainer Board (Kaycomp II or Flight Model).

PROCEDURE

All the programs in the exercises have to be written and assembled using the EASy68K simulator. To observe the results, the programs have to be downloaded and executed on the MC68000 CPU Trainer Board. The contents of the affected registers and memory locations have to be examined through single stepping or breakpoints setting.

EXERCISES

1. Write a program that will store $F2, $1478 and $2B1478FF in three data registers. Execute the program and explain the differences in the results.

2. Write a program that will store $FF45E367 in a data register and copy the **low word** in D0 and the **long word** in D2. Execute the program and comment on the results.

3. Write and execute a program that will store addresses $6743 and $400650 in two address registers. Explain the differences in the results.

4. Write a program that will store $5489A2B3 in a data register and copy the **low word** in memory location $400450 and the high word in memory location $400600. Execute the program and comment on the results.

5. Locations $400440 and $400441 contain a word number. Write a program which will load the contents of these memory locations in a data register and copy the content of these memory into memory locations $400450 and $400451 respectively using the following modes of addressing:
   a) Absolute addressing mode
   b) Indirect addressing mode

LAB 2: MC68000 CPU PROGRAMMING
DATA MANIPULATION INSTRUCTIONS

OBJECTIVES

At the end of the laboratory works, you should be able to write simple assembly language programs for the MC68000 CPU using data manipulation instructions.

INTRODUCTION

The capability of a CPU depends on its data manipulation instructions which include the arithmetic, logical, shift, rotate, the bit manipulation and BCD instructions. Some of the arithmetic instructions are ADD, ADDI, SUB, SUBI, CLR, CMP, DIVS, DIVU, MULS, MULU and NEG. The logical instructions are AND, ANDI, OR, ORI, EOR, EORI and NOT.

Some of the shift and rotate instructions are ASL, ASR, LSL, LSR, ROL and ROR. Bit manipulation deals with bit operations, and the instructions are BCHG, BCLR, BSET and BTST. For the operation involving BCD numbers, the MC68000 provides ABCD, NBCD and SBCD instructions.

EQUIPMENTS

1. A Personal computer installed with the MC68000 CPU Editor/Assembler and Simulator software, EASy68K.
2. MC68000 CPU Trainer Board (Kaycomp II or Flight Model).

PROCEDURE

All the programs in the exercises have to be written and assembled using the EASy68K simulator. To observe the results, the programs have to be downloaded and executed on the MC68000 CPU Trainer Board. The contents of the affected registers and memory locations have to be examined through single stepping or breakpoints setting.

EXERCISES

1. Write and execute a program that add the four 8-bit numbers $13, $66, $B9 and $9C using:
   - i). Byte addition.
   - ii). Word addition.

   The result must be stored in memory location $400500. Comment on your results.

2. Write and execute a program that will subtract $FF15 from $FE4A using
   - i). Word subtraction
   - ii). Long word subtraction.

   Store the results in memory location $400500 and explain the difference in the results.

3. Write and execute a program that divides $0008B120 by $0000018C using DIVU. Explain on the result obtained.

4. Examine the function of the MULS and MULU instructions by executing a program that multiply the numbers $FFF0 and $FFF6 using both instructions. Explain the difference in the results obtained.

5. a) Write and execute a program that logically AND the **low word** of the numbers $4865AB5E and $A32C4B9D. Explain on the result obtained.
   b) Modify Part 7 a) to perform a logical OR of the **high word** of the two numbers. Execute the program and explain on the results obtained.

6. Examine the difference in the results obtained after the ASL and LSL operations performed three times on the **low word** of the number $4582A2C4. Explain the result of each instruction.

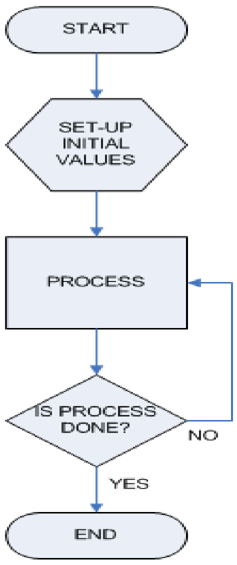
LAB 3: MC68000 CPU PROGRAMMING
PROGRAM LOOPS AND SUBROUTINES

OBJECTIVES

At the end of the laboratory work, you should be able to write the MC68000 assembly language program using looping technique and subroutines.

INTRODUCTION

The programs considered in the previous laboratory exercises so far are all the straight line, sequential variety. In practice, programs almost invariably involve the repetition of sections of the routine in order to achieve a solution or complete a control process. Such repetitions are known as loops and it is in these that the real power of microprocessor lies. The flowchart of generalized loop procedure is shown.



The loop normally requires setting up initial values for use by the process and also by the loop, for continuing and terminating the loop. The number of loops executed is stored in the loop counter.
The process is the “compound instructions” that will be executed repeatedly when the loop repeats itself. Extra care must be exercised to formulate these instructions.
The instruction used to test for terminating the loop is called conditional instructions, and is of the form Bcc, where B stands for branch and cc is the condition. Some commonly used conditional instructions are BNE, BEQ, BGE, BGT, BLE and BLT.

Subroutine is a technique of writing programs so that it gives the advantages of using less memory space for programs and the modular program structure which leads to easier debugging.

EQUIPMENTS

1. A Personal computer installed with the MC68000 CPU Editor/Assembler and Simulator software, EASy68K.
2. MC68000 CPU Trainer Board (Kaycomp II or Flight Model).

PROCEDURE

All the programs in the exercises have to be written and assembled using the EASy68K simulator. To observe the results, the programs have to be downloaded and executed on the MC68000 CPU Trainer Board. The contents of the affected registers and memory locations have to be examined through single stepping or breakpoints setting.

EXERCISES

1. Write and execute a program that copy twelve bytes of data starting from memory locations $400470 to memory locations starting $400450H.

2. Write and execute a program that place the larger of the contents of memory locations $400460 and $400462 into memory location $400464.

3. Three hexadecimal numbers are stored in memory locations $400700, $400701 and $400702. Sort these numbers into ascending order with the smallest number in memory location $400700.

4. Write a program, which inspect the contents of memory location $400440 and, if the contents are greater than $20, subtract the value with $11 and store the result in memory location $400540. Otherwise, add it with $10 and store the result in location $400550.

5. Twelve 8-bit numbers are stored in memory locations starting with $400500. Add the content of these memory locations using **indirect addressing with post increment and loop**. Execute the program and explain on the result obtained.

6. Write and execute a program that compute the area of two circles whose diameters are stored in memory locations $400550 and $400560, and save the area of these circles in memory locations $400570 and $400580. The calculation process of the area must be done in a subroutine.

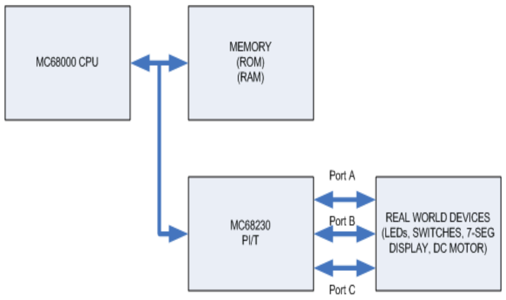
LAB 4: MC68000 CPU PROGRAMMING
INPUT/OUTPUT PROGRAMMING

OBJECTIVES

At the end of the laboratory work, you should be able to write the MC68000 input/output assembly language programs for the MC68230 Parallel Interface/Timer (PI/T) device.

INTRODUCTION

The MC68000 CPU uses the Parallel Interface/Timer (PI/T) chip MC68230 to communicate with the external devices. The PI/T allows 8-bit data to be transferred in and out simultaneously. Before input/output operations can be carried out, the PI/T must first be initialized.


MC68000 CPU
MEMORY (ROM) (RAM)
MC68230 PI/T
Port A
Port B
Port C
REAL WORLD DEVICES (LEDs, SWITCHES, 7-SEG DISPLAY, DC MOTOR)

The PI/T has 32 of registers, but for simple input/output programming, only the Data Direction Register (DDR) and Data Register (DR) are used. There are three ports on the PI/T; Port A, Port B and Port C where each port has 8 lines to output an 8 bit data at a time.

EQUIPMENTS

The assembly language programs can be written, assembled and executed using MC68000 CPU Trainer Board (ABLELOGIC). The Trainer Board comes complete with Application Board which enables all the programs to be tested with the appropriate devices. Table 1 shows some of the real world devices on the Application Board and their connections to Port A and B lines.

Table 1 The connections of the input and output devices to the ports.

Devices	**Port Connections**	
	Port A Addresses : DDR $800005 DR $800011 Data registers: D7,D6,D5,D4,D3,D2,D1,D0	**Port B** Addresses : DDR $800007 DR $800011 Data registers: D7,D6,D5,D4,D3,D2,D1,D0
7-Seg. Units dp, g, f, e, d, c, b, a		D7,D6,D5,D4,D3,D2,D1,D0
7-Seg. Units Enable	D3,D2,D1,D0	
LEDs		D7,D6,D5,D4,D3,D2,D1,D0
DAC		D7,D6,D5,D4,D3,D2,D1,D0
Comparator Output	D4	
Switches	D7,D6,D5,D4,D3,D2,D1,D0	
DC Motor		D6, D7
Heater		D5

PROCEDURE

All the programs in the exercises have to be written and assembled using the EASy68K simulator. To observe the results, the programs have to be downloaded and executed on the MC68000 CPU Trainer Board. The application board has to be connected to the CPU Trainer Board before executing the program. The contents of the affected registers and memory locations have to be examined through single stepping and breakpoints setting if necessary.

EXERCISES

1. Write and execute a program that turns on LED that connected to Port B1, Port B3, Port B4 and Port B7.

2. Write and execute a program that reads the condition of 8 switches connected at Port A and store the data in a data register.

3. a) Write and execute a program that reads the condition of the switches at Port A, stores it in memory location $400460 and turns the LED on at Port B.
   b) Modify this program so that it repeats reading and displaying the data, and it will stop when data is greater than $B7.

4. a) Write, assemble and execute a program that output a counting down in binary (from $FF to $00) at Port B on the DAC with a delay of 2s for every count. The voltage output from the DAC should be observed using a voltmeter or an oscilloscope.

   b) Modify the above program so that it generates a square wave at the DAC output.

5. Write and execute a program that displays the numbers 27 and 65 alternately on the 7-segment units with 3s delay in between.