ECE 441 MONITOR DESIGN PROJECT SPRING 2015

Project Demo & Full Report Due Date

Section 1 : Tuesday, April 28th 2015

Section 2 & 3: Wednesday, April 29th 2015

Design Objective:

Since we have learned the different aspects of hardware design such as memory and I/O design with a MC68000 processor, the goal of this Design Project is to build up a Monitor program with MC68000 assembly language. Your Monitor program should be able to perform basic debugger functions (memory display, memory sort, memory change, block fill, block search, block move), and ability to handle exceptions (bus, address, illegal instruction, etc.). You are required to design and create the list of necessary Monitor commands.

It is required that you divide your project into four components:

1. **Command Interpreter**

   The Monitor recognizes which command is entered and branches to subroutine for executing this command. If there is no match, it displays an error message. You must include adequate error handling and error checks in the Command Interpreter. See Appendix I for more detailed description.

2. **Debugger Commands**

   You must define appropriate commands (*all 12 commands described in Page 2 & 3 and* ***TWO additional instructions of your own*** *for the full Monitor program implementation*) and provide adequate descriptions for including these commands. You must also describe the basis for your selection of the commands and provide minimum three references for the Monitor program. The Monitor must be able to execute your selection of commands after the prompt (e.g. "MONITOR441>") and terminated with carriage return, <CR>. The group of commands must deal with memory operations including memory display, modify, move, search and memory testing.

3. **Exception Handlers**

   For the **Exception Handler**, the Monitor program must be able to handle *ALL* system exceptions of MC68000. They are:

   - Bus Error Exception **(must include SSW, BA, IR Outputs. Refer Lab 3)**
   - Address Error Exception **(must include SSW, BA, IR Outputs. Refer Lab 3)**
   - Illegal Instruction Exception
   - Privilege Violation Exception
   - Divide by Zero Exception
   - Check Instruction Exception
   - Line A and Line F Emulator Exception

   *** For those who are using EASy68K, you must enable "Enable Exception" option in SIM68K before running*

4. **User Instruction Manual**

   For the **User Instruction Manual**, the Monitor should include a help screen for every command and their syntax such as the required parameters and operands.

Project Details:

Debugger Commands

In this part, the monitor must be able to execute the following commands after prompt " MONITOR441>" and terminated with carriage return, <CR>.

You MUST use the command names as specified. Also, you MUST follow the syntax described in this project description.

HELP (Help)

- The Help command must display all available commands and description of their usage

MDSP (Memory Display)

- The MDSP (Memory Display) command outputs the address and memory contents from <address1> to <address2>.
- The MDSP (Memory Display) command also outputs the address and memory contents from <address1> to <address1 + 16bytes>.

HXDEC (Conversion)

- Converts a hexadecimal number to decimal number

Note: You cannot use Trap #14 (SANPER) or #15 (EASy68K) functions for conversion purposes.

SORTW (Sort)

- The SORT command sorts a block of memory. The starting address and the ending address of the memory block are specified in the command. The order (A or D) specifies whether the list is sorted in ascending or descending order. (The size of the data to be sorted is a word)

ORDER	DESCRIPTION
;A	Ascending order
;D	Descending order
-(default)	Descending order

MM (Memory Modify)

- MM (Memory Modify) command is used to display memory and, as required, modify data or enter new data. The size (byte, word, long word) controls the number of bytes displayed for each address.

SIZE	DESCRIPTION
-(default)	Displays one byte
;W	Displays one word (2 bytes)
;L	Displays one long word (4 bytes)

MS (Memory Set)

- The Memory Set (MS) command alters memory by setting data into the address specified. The data can take the form of ASCII string or hexadecimal data.

BF (Block Fill)

- The Block Fill (BF) command fills memory starting with the word boundary <address1> through <address2>. Both <address1> and <address2> must be even addresses. This command only fills with a word-size (two-byte) data pattern. If an entire word size data pattern is not entered, the pattern is right justified and leading zeros are inserted.

BMOV (Block Move)

- The Block Move (BMOV) command is used to move (duplicate) blocks of memory from one area to another.

BTST (Block Test)

- The Block Test (BT) command is a destructive test of a block of memory beginning at <address1> through <address2>. If this test runs to completion without detecting errors, and display a message that no error was detected.
  If memory problems are found, a message is displayed indicating the address, the data stored, and the data read of the failing memory.

BSCH (Block Search)

- The BSCH (Block Search) command is used to search a literal string in a memory block starting at <address1> through <address2> both inclusive. In BSCH command, if search finds matching data, the **data** and **address(es)** must be displayed.

GO (Execute Program)

- The GO command causes the target program to execute (free run in real time)

DF (Display Formatted Registers)

- The Display Formatted Registers (DF) command is used to display the MC68000 processor registers. This command should display current PC, SR, US, SS and D, A registers. Note that since your program will be using D and A registers, those changed values should not be shown to the user. Thus, you must save all D and A register values before any of your subroutines.

In addition to above 12 commands, you MUST define 2 additional commands for this project. You are required to implement and include detailed descriptions of your 2 additional commands.

Design Constraints:

The program and look up tables have the following **practical design constraints**:

- Entire code must be smaller than 3K size starting from $1000 (including look-up tables for help and error messages).
- 1 K stack size residing in memory locations $3000 and up
- Include any relevant I/O Trap #14(for SANPER) or #15(for EASy68K) routines in code

- ***<u>YOU MAY NOT USE MACROS AT ALL IN THIS PROJECT! You will get ZERO POINTS if found.</u>***
- Erroneous inputs should not kill program but the number of errors statements should be minimum.

Requirements and Report Contents:

- ***<u>This project should be finished individually and independently</u>***. However, you are encouraged to discuss with your classmates. DO NOT share codes with others whatsoever. If plagiarism found,
- The firmware has to be command driven and properly organized.
- The software of project has to be demonstrated on the SANPER-1 ELU or EASy68K individually to the TA. (*Time slots will be announced later*)
- Instructions for each command must be fully described in the project report. ERROR messages must guide the user for correct usage of the commands.
- **A Project Report must be submitted in hard copy (<u>WILL NOT accept any electronic copies</u>)** that would include: (given template is for your guide)
    - Table of contents and Abstract
    - Clear and concise description of the implementation of commands
    - A separate section for the flowcharts/algorithms used in
        a. Command Interpreter
        b. Individual Commands
    - Code listings with global and local comments
    - A quick manual for command usage and monitor
    - A separate section where you highlight the engineering and design challenges and methods to overcome these challenges
    - A separate section discussing the expansion of the existing Monitor program for more advanced implementations. How would this project help if you are developing an OS for an embedded system using MC68000? Provide an example and discuss the features for this.
    - Conclusion and References

Grading Policy:

- ❑ Demonstration of the Project
    - ❑ Must submit your code before your demonstration <u>(submit to Won-Jae Yi via e-mail "wyi3@iit.edu")</u>
    - ❑ Testing and evaluation of the command interpreter
    - ❑ Functional evaluation of each command
    - ❑ Evaluation of exception handling
    - ❑ Clarity of presentation, syntax error handling and help messages
- ❑ Final Report
    - ❑ Must include all the required sections listed above

Note: 1. Demonstrate whatever you did and show your work. Without any demo, a report is not acceptable.
2. Any modification on the command name is not acceptable.