ECE 441
Monitor Project

Outline

- **Introduction**
- **Command Interpreter**
- **Debugger Commands**
- **Exception Handlers**
- **User Instruction Manual**
- **Guidelines and Helpful Hints**
- **Project Report Template**
- **Project Demonstration in EASy68K**

Introduction

- The goal of the project is to build up a monitor program.
- The monitor should be able to perform basic debugger functions, such as memory display, memory sort, memory change, block fill, block search, block move and also, it can deal with exceptions.

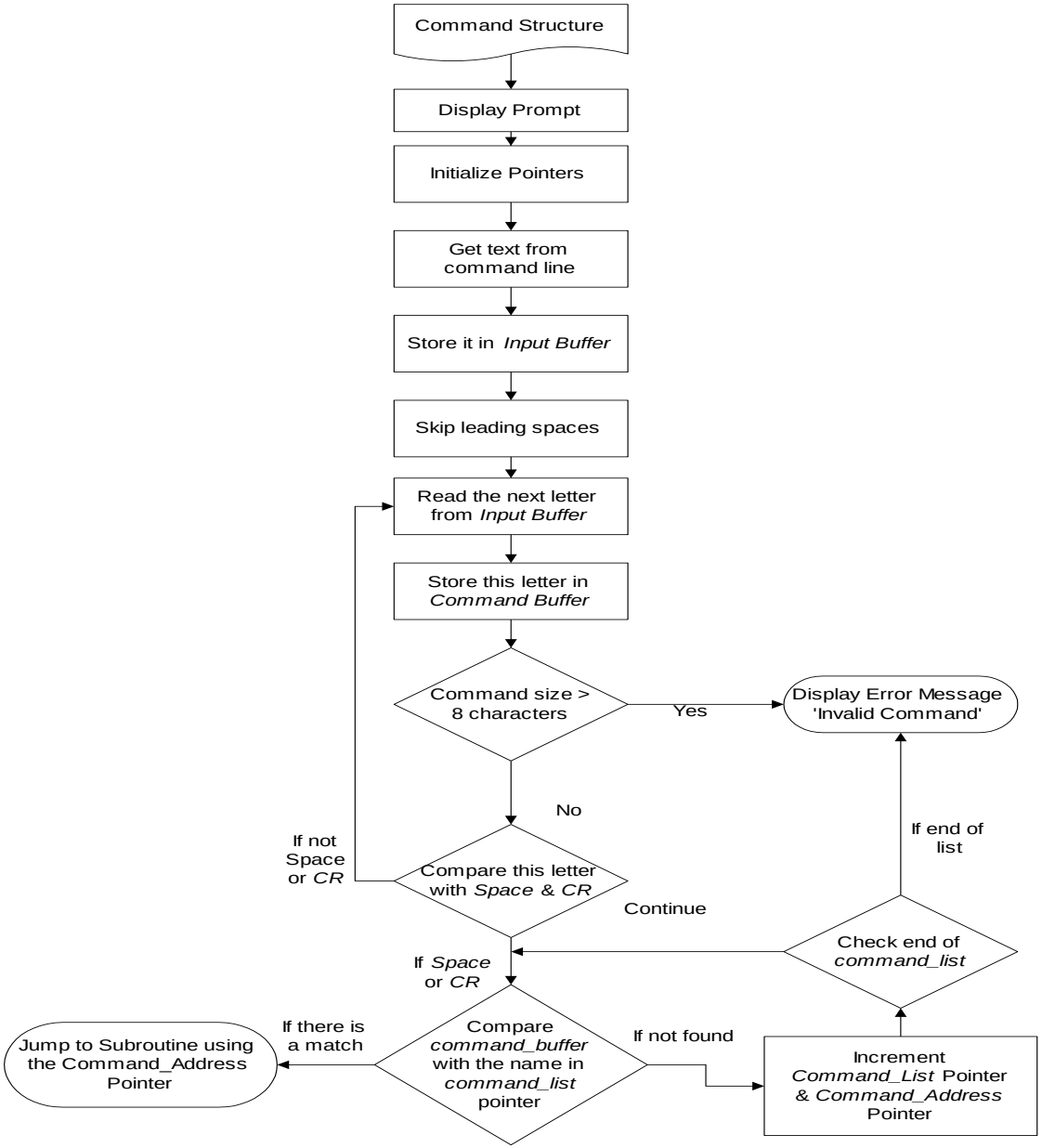
Introduction

- 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/#15 routines in code (SANPER or EASy68K)
- Erroneous inputs should not kill program but the number of errors statements should be minimum.

Command Interpreter

- The monitor recognizes which command is entered and then branches to subroutine for executing this command. And when there is no match, it displays error message.
  - COMP_TABL is a pointer on the command names.
  - COMP_ADDR is a pointer on the addresses of the subroutines.
  - INPUT is a buffer where the input command with the arguments is stored.

Command Interpreter Flowchart



Debugger Commands

- In the **debugger commands** part,
  - Must define appropriate commands
    - (all 12 commands and 2 additional commands for the full monitor program implementation)
  - Provide adequate descriptions for including these commands.
  - Must describe the basis for your selection of the commands and provide minimum 3 references for the monitor program.
- The monitor must be able to execute your selection of commands after prompt “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.

Debugger Commands

- There will be all 12 Commands and 2 Additional Debugger Commands
- Look for TUTOR user guide for more information
- Full description of the functionality for each command
- Detailed explanation of the command usage
  - Each command should have algorithm && flowchart
- Sample output for each command

Example (MDSP)

Memory Display

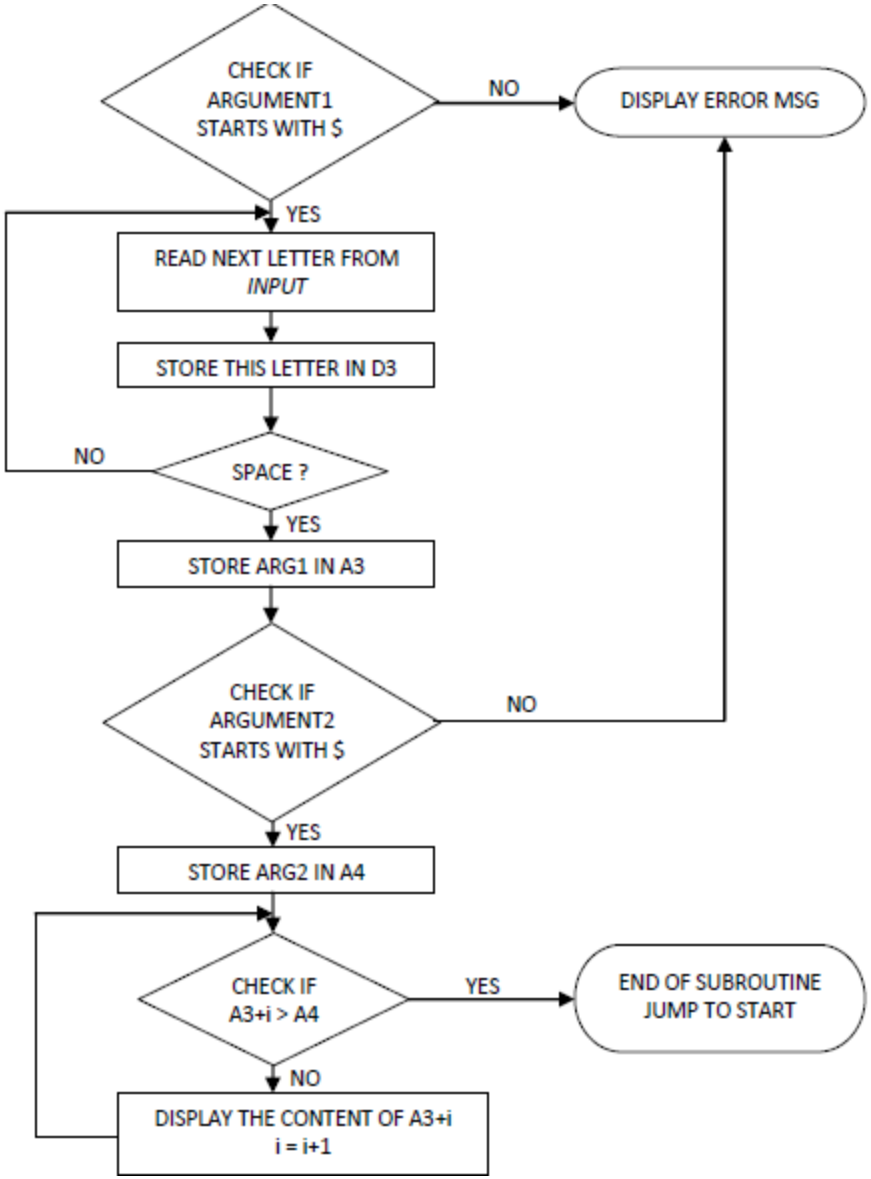
- The MDSP (Memory Display) command outputs the address and memory contents from <address1> to <address2> :

- MONITOR441> MDSP <address1> <address2>

- Example…..
- MONITOR441>MDSP $908 $90A <CR>
- 908 06
- 909 FF
- 90A C2

Example (MDSP)

Memory Display



Example (MDSP)
Memory Display

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

- MONITOR441> MDSP <address1>
- Example…..
- MONITOR441>MDSP $900<CR>
- 900 06
- 901 FF
- …. ….
- 915 00

Exception Handlers

In the **exception handler** part, the monitor must be able to handle the different system exceptions of MC68000. *(Use Lab 3 as reference)*

YOU MUST ENABLE EXCEPTION in EASY68K

(Menu -> Options -> Enable Exceptions)

- Bus & Address Error Exception
  - This exception handler should also display the contents of Supervisor Status Word, Bus Address and Instruction register all in a single line with spaces in between them
- Illegal Instruction Exception
- Privilege Violation Exception
- Divide by Zero Exception
- CHK Instruction Exception
- Line A and Line F Emulators

Exception Handlers

In order to handle exception we have modified the exception vector table:

- MOVE.L #BUS_ERR,$8
- MOVE.L #ADS_ERR,$C
- MOVE.L #IL_INST,$10
- MOVE.L #DIV_Z,$14
- MOVE.L #CHK_INST, $18
- MOVE.L #PRI_VIO,$20
- MOVE.L #LINE_A,$28
- MOVE.L #LINE_F,$2C

Exception Handlers

- For each exception we need to display the value of the registers, so we store all the registers in the stack starting at address $3000
- Then we display all the registers, we restore the values and we jump back to our monitor program.
- For the bus error and address error routines, we need also to display the content of the Supervisor Status Word, Bus Address and Instruction register all in a single line with spaces between them.
  - This is done by reading the content of A7.

Exception Handlers

```
ORG $900
MOVE.B   $FFFFF,D0
MOVE.B   #228,D7
TRAP     #14
```

```
MONITOR441> GO $900
Bus Error Exception
1035 000FFFFF 1039
D0=00000001 D1=30303031 D2=00000012 D3=00000024
D4=00000035 D5=00000030 D6=00000900 D7=0000FFF1
A0=00001014 A1=00001014 A2=0000153B A3=00000900
A4=00002CA2 A5=0000100D A6=00001014 A7=0000076C

Line F Emulator
D0=0000076C D1=30373643 D2=00000038 D3=00000024
D4=00000035 D5=00000030 D6=00000900 D7=0000FFF8
A0=00003040 A1=00001014 A2=0000153B A3=00000900
A4=00002CA2 A5=00003000 A6=00003000 A7=00000772

MONITOR441>
```

User Instruction Manual

In the **instruction manual** part, the monitor program should include a help screen for every command and their syntax such as the required parameters and operands

User Instruction Manual

```
MONITOR441> HELP
HELP: Displays this message

MDSP: Outputs Address And Memory Contents
MDSP <address1> <address2> eg: MDSP $908 $90A<CR>

MCHG: Modifies Data In Memory
Default: Displays One Byte
;W: Displays One Word
;L: Displqys One Long Word
MCHG <address>[;size] eg: MCHG $904;W<CR>

SORTW: Sorts A Block Of Memory
Default: Descending Order
;A: Ascending Order
;D: Descending Order
SORTW <address1> <address2> [;order] eg: SORTW $904 $90E ;A<CR>

BFILL: Fills A Block Of Memory
BFILL <address1> <address2> <word> eg: BFILL $904 $908 475A<CR>

BSCH: Searches A Literal String In The Memory
BSCH <address1> <address2> "literal string" eg: BSCH $900 $910 "MATCH"<CR>

BMOV: Moves A Block Of Memory To Another Area
BMOV <address1> <address2> <address3> eg: BMOV $908 $90B $909<CR>

HXDEC: Converts A Hex Number To A Dec Number
HXDEC <hex number> eg: HXDEC $40<CR>

GO: Starts Execution From Given Address
GO <address> eg: GO $900<CR>

EXIT: Exit the monitor program eg: EXIT<CR>
```

Guidelines and Helpful Hints

First step in the monitor program is to initialize the system constants.

- Assign memory addresses to STACK,

  Example:

```
STACK       EQU $XXXXXX
.
.
.
```

- Initialize all necessary look-up tables required.

Guidelines and Helpful Hints

- Create the Command Table: Each command should take two long-word size memory spaces and end with '*space*' character. This will mark the end of the command.
- First part in the table is the *command names*, second part is the *command addresses*.

```
                ORG $XXXX
COM_TABL        DC.B    'HELP ‘    ; String ‘HELP’ stored in memory (ASCII)
                DC.B    ‘MDSP ‘
                DC.B    'MTST ‘
COM_ADDR        DC.W    HELP       ;HELP is the address for HELP command subroutine
                DC.W    MDSP
                DC.W    MTST
                          .
                          .
                ORG $XXXX
HELP            Put your code here
                Return to your monitor program
MDSP            Put your code here
                Return to your monitor program
IMMTST          Put your code here
Return to your monitor program
```

Guidelines and Helpful Hints

- Second step is to setup exception vector table. You will develop your own exception handler routines for most of these exceptions.

```
        ORG         $XXXX
MOVE.L  #STACK,     $0
MOVE.L  #BUS_ERR,   $8
MOVE.L  #ADRS_ERR,  $C
MOVE.L  #ILL_INST,  $10
MOVE.L  #DIV_ZERO,  $14
MOVE.L  #CHK_INST,  $18
MOVE.L  #PRIV_VIOL, $20
MOVE.L  #LINE_A,    $28
MOVE.L  #LINE_F,    $2C
```

Guidelines and Helpful Hints

- Third step is to write the code for the *Command Interpreter*:
- Command Interpreter recognizes which command is entered. Then, branches to subroutine for executing this command. It compares the input for all the commands in the command table and when there is no match, it displays error message: "Invalid command".
- Error check for *arguments* is done inside the command subroutine.
- If there is an error, each command subroutine displays error message: "Invalid command"
- Note that command subroutines should branch back to the start of monitor program after execution.

Guidelines and Helpful Hints

- You may use two pointers for implementing command table:
- ***Command_list pointer*** points to the command names in ASCII format. These names are used to match the input text.
- ***Command_address pointer*** points at the actual address of the command subroutines in the table.
- While doing a search in the table, all the letters should match and they should end with either '*Space*' or '*CR*' character. Incomplete command names or redundant letters in the command name are not acceptable and will cause '*Invalid Command*' error
- Use **JSR** for jumping to the command instruction, **RTS** after when done with the command instruction.

Project Report Template

- Please follow the given report template
- Will distribute to your e-mail address
- If you do not receive this template file, please send us an e-mail!

- **Start your final project now!!**