CEG 3136 – COMPUTER ARCHITECTURE II

LAB 1

INTRODUCTION TO MICROPROCESSOR PROGRAMMING

STUDENT NAME:

1. HAZIQ HAFIZI BIN MD YUSOF (8346453)
2. HANIF BIN ARIFFIN (8346480)

DATE: SEPTEMBER 18,2017

TA: Anas Alakhras

**OBJECTIVES**

MAIN OBJECTIVE:

* Become familiar with the Debug-12 Monitor
* Assemble and run assembler code
* Learn basic principle of debugging
* Apply different addressing modes
* Design and implement a simple software module

SUB-OBJECTIVE:

* Get to know the shape of the Dragon 12 and the lab for CEG 3136 works

**EQUIPMENTS AND COMPONENTS**

* Dragon 12 Plus Trainer
* Windows PC
* MiniIDE

**SOFTWARE / HARDWARE DESIGN**

**Part 1**

;-----------------

; LAB 1 Part B

;-----------------

 ORG $2000

LDD #$3A

LDX $EE86

JSR 0,X

LDX $EE84

JSR 0,X

LDAA #3

PSHA

  PSHB

LDD #$20

LDX $EE86

JSR 0,X

PULB

LDX $EE86

JSR 0,X

PULA

DECA

BNE $200F

SWI

Original Program

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Content** | **Instruction** | **Description** |
| 2000 | CC003A | LDD #$3A | Load ASCII code for “:” |
| 2003 | FEEE86 | LDX $EE86 | Load the vector for *putchar* routine |
| 2006 | 1500 | JSR 0, X | Print what’s in B on terminal |
| 2008 | FEEE84 | LDX $EE84 | Load the vector for *getchar* routine |
| 200B | 1500 | JSR 0, X | Get a new character in B |
| 200D | 8603 | LDAA #3 | Initialize loop counter |
| 200F | 35 | PSHA | Save the counter on stack |
| 2010 | 37 | PSHB | Save contents of B on stack |
| 2011 | CC0020 | LDD #$20 | Load B with a space |
| 2014 | FEEE86 | LDX $EE86 | Load the vector for *putchar* routine |
| 2017 | 1500 | JSR 0, X | Print it on terminal |
| 2019 | 33 | PULB | Get original character |
| 201A | FEEE86 | LDX $EE86 | Load the vector for *putchar* routine |
| 201D | 1500 | JSR 0, X | Print it on terminal |
| 201F | 32 | PULA | Retrieve the counter |
| 2020 | 43 | DECA | Decrement loop counter |
| 2021 | 26EC | BNE $200F | If counter <> 0, repeat |
| 2023 | 3F | SWI | Return to the monitor |

Modified Program

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Content** | **Instruction** | **Description** |
| 2000 | CC003A | LDD #$3E | Load ASCII code for “>” |
| 2003 | FEEE86 | LDX $EE86 | Load the vector for *putchar* routine |
| 2006 | 1500 | JSR 0, X | Print what’s in B on terminal |
| 2008 | FEEE84 | LDX $EE84 | Load the vector for *getchar* routine |
| 200B | 1500 | JSR 0, X | Get a new character in B |
| 200D | 8603 | LDAA #15 | Initialize loop counter |
| 200F | 35 | PSHA | Save the counter on stack |
| 2010 | 37 | PSHB | Save contents of B on stack |
| 2011 | CC0020 | LDD #$3B | Load B with a “;” |
| 2014 | FEEE86 | LDX $EE86 | Load the vector for *putchar* routine |
| 2017 | 1500 | JSR 0, X | Print it on terminal |
| 2019 | 33 | PULB | Get original character |
| 201A | FEEE86 | LDX $EE86 | Load the vector for *putchar* routine |
| 201D | 1500 | JSR 0, X | Print it on terminal |
| 201F | 32 | PULA | Retrieve the counter |
| 2020 | 43 | DECA | Decrement loop counter |
| 2021 | 26EC | BNE $200F | If counter <> 0, repeat |
| 2023 | 3F | SWI | Return to the monitor |

B) LDX $EE86  = Direct addressing with hexadecimal number

JSR 0, X = Index addressing

BNE $200F  = Direct addressing with hexadecimal

C) The program loops three times. Each loop, the program will get an input from user. The input will be then display on the monitor and then concatenated with space. The order of the ‘instructions’ in C is no the same like in assembly.

void lab-1-part2(){

std::cout << “:” << std::endl; //print “:” to the terminal

char b; //saving the content of B to the stack

std::cin >> b; //expect input from user

int a = 3; //save the content of A to the stack

while (a-- != 0){

std::cout << “ “;

std::cout << b;

}

}

D) Change LDD #$3A to LDD #$3E to change “:” to “>”

Change LDD #$20 to LDD #$3B

E) Change LDAA #3 to LDAA #15

**PART 2**

From debugging, we found that in the armed.asm, an opcode beq cdv\_while in method does not branch to cdv\_while, thus it does not check the code entered by user to disarm or arm the alarm. This is caused by the opcode check the Z flag and does not branch to cdv\_while because Z = 0. Thus, to fix this bug, we changed beq cdv\_while to bne cdv\_while which cause the behavior of the program change as if it checks the Z=0, it will branch to cdv\_while to keep looking if the code entered by the user is in the array.

**PART 3**

The core clock is set 24Mhz thus 1 cycle will be 4 2/3ns. We need 24,000 cycles to create a 1ms delay. Thus, we use 4 NOP , 1 DEX and 1 BNE and repeat for 3000 times. NOP will take 1 cycle, BNE will take 3 cycles and DEX will take 1 cycle thus everything will take 8 cycles and if it is repeated 3000 times, it will be 24,000 cycles and thus create 1ms delay.

**Conclusion**

Lab 1 has succeeded although he faced a lot of problem. We had some hardware issue as the Dragon 12 could not run alarmsimul.asm at all. After that, we able to run the program but the terminal did not print the menu properly but it works on our friend’s board. Thus, we have issue to debug the program properly. However, we able to complete the lab on time.