Lab 1

ENSC 332

Dan Hendry (30113878)
Nicholas Flandin (301062840)

Simon Fraser University
School of Engineering Science

June 14, 2010
Course Instructor: Professor M. Moallem

1 Introduction

This lab explores basic assembly language programming for the HCS12 micro-controller using the CodeWarrior Integrated development environment (IDE). Over the course of two activities, two simple programs were written and tested, first using the CodeWarrior simulator and then by downloading them to the development board. Registers and memory locations were examined in both testing phases to verify the programs were behaving as expected.

2 Activity 1

The first activity involved writing an assembly program which transfered a specific value to eight predefined memory locations. The template given in the lab instructions was basis for creating the program. A simplistic program was written which loaded the value to be copied to RAM into accumulator A. It then used eight separate assembly instructions to write to each memory address. After executing the eight store instructions, the program entered an infinite loop, effectively ending execution. A potentially more efficient and elegant solution would used a counter use a single store command repeated eight times. Regardless, the program performed as expected and using the debugging environment, the value of the RAM locations was seen to have the desired and expected value.

2.1 Code

```
; Assembly Language Program
        ABSENTRY Entry ; absolute assembly application entry point
; Include derivative-specific definitions
        INCLUDE 'mc9s12dg256.inc'
ROMStart EQU $4000 ; absolute address to place my code/constant data
; $4000 is where code ROM starts for 9s12dx256 up to $7fff
; variable/data section
        ORG RAMStart ; RAMStart is defined in mc9s12dj256.inc as $1000
; Insert here your data definition.
Counter DS.W 1 ; set aside 1 word for counter in RAM
FiboRes DS.W 1
; code section
        ORG ROMStart
Entry:
        LDAA #$99 ; Load the value to be stored
        STAA $1220 ; Store it to 8 ram locations
        STAA $1221
        STAA $1222
        STAA $1223
        STAA $1224
        STAA $1225
        STAA $1226
        STAA $1227
HERE JMP HERE
```

```
        END ; if system has monitor program END can be removed
;***********************************************
;* Where to go when reset key is pressed *
;***********************************************
        ORG $FFFE
        DC.W Entry ; Reset Vector
```

3 Activity 2

The second activity involved writing an assembly program which added five numbers and stored the result in register A. The *ADDA* instruction was used to sum a set of predefined values in-place in accumulator A. The program terminated by entering an infinite loop. Stepping through the program line by line using the debugger, the final value of accumulator A was verified.

3.1 Code

```
; Example Assembly Language Program
        ABSENTRY Entry ; absolute assembly application entry point
; Include derivative-specific definitions
        INCLUDE 'mc9s12dg256.inc'
ROMStart EQU $4000 ; absolute address to place my code/constant data
; $4000 is where code ROM starts for 9s12dx256 up to $7fff
; variable/data section
        ORG RAMStart ; RAMStart is defined in mc9s12dj256.inc as $1000
; Insert here your data definition.
Counter DS.W 1 ; set aside 1 word for counter in RAM
FiboRes DS.W 1
; code section
        ORG ROMStart
Entry:
        LDAA #$42 ; First number to be summed
        ADDA #$D ; Second through fifth numbers
        ADDA #$F
        ADDA #$D
        ADDA #$F
HERE JMP HERE
        END ; if system has monitor program END can be removed
;*****************************************
;* Where to go when reset key is pressed *
;*****************************************
        ORG $FFFE
        DC.W Entry ; Reset Vector
```

4 Questions

4.1 Registers

Information about each register is given in table 1 below.

Table 1: Register

	PC	A	B	X	Y	PORTB
Register size (bits)	16	8	8	16	16	8
Largest Value (Decimal)	65535	255	255	65535	65535	255
Largest Value (Hex)	FFFF	FF	FF	FFFF	FFFF	FF

4.2 Files and Compilation

The process of translating the assembly program into the binary image which is loaded onto the processor involves the creation of a number of different files. These files, and the program responsible for their generation, are listed below.

.asm - This file is the assembly program, written to perform a specific task. It is created by the programmer using the CodeWarrior IDE.

.lst - The .lst file is generated by the assembler and lists any errors which were detected in the .asm file.

.obj - The .obj file is also generated by the assembler. It contains the machine code.

.s19 - This file is generated by the linker from a given set of object files. It is a hex file containing the data which is to be downloaded to the processor.