Lab 2

ENSC 332

Dan Hendry (30113878)
Nicholas Flandin (301062840)

Simon Fraser University
School of Engineering Science

July 2, 2010
Course Instructor: Professor M. Moallem

1 Introduction

This lab involved the creation of three assembly programs to examine the call and branch instructions. A major application of these instructions is in the creation of time delay loops. The program template for the first lab was used for each activity.

2 Activity 1

The first activity required the creation of an assembly program which added the number three to accumulator A fifteen times before exiting. The X index register was used as a counter. After each addition, it was decremented and if it was not equal to zero, a loop to perform the desired action (adding 3 to register A) was repeated.

The program was simulated in CodeWarrior and was seen to behave as expected. At termination, accumulator A had a value of 0x2D and register X a value of 0x0. Partial code for this activity is shown below. This was run within the template provided in the first lab.

2.1 Code

```
EntryA1: ;       Entry for the first assignment
        CLRA ; Clear reg a
        LDX #$000F ; Load the number of times to count in X

Loop: ; Loop 15 times
        ADDA #$03
        DEX
        BNE Loop

        JMP HERE ; End - go into an infinite loop
```

3 Activity 2

The second activity required the creation of a generic program capable of processing more that 16 million iterations. Both index registers, X and Y, were used as decrementing counters. Since each is 16 bits, a loop capable of handling 2^{32} (approximately 4.3 billion) iterations was created. The code shown below will iterate 6 times and perform a NOP on each iteration. The program was simulated using CodeWarrior and was seen to behave as expected, performing a NOP six times before terminating.

3.1 Code

The number of iterations which will be performed is given by the values of X and Y multiplied before the program executes. In this case, 2*3=6 iterations will be performed.

```
EntryA2: ;Entry point for the second activity
        CLRA
        LDX #$0002 ; Outer counter

LoopOuter:
        LDY #$0003 ; Inner counter
LoopInner:

        ;Some action
    ;This section will be repeated
        NOP

        DEY
        BNE LoopInner
        DEX
        BNE LoopOuter

        ;End
        ;RTS ; Uncomment when being used for activity 3
        JMP HERE
```

4 Activity 3

The third activity required the creation of an assembly program to count from $0 to $A twice, storing the current value in accumulator B. A short delay between incrementing A was required and implemented using the program from activity 2 as a subroutine. For testing, the value of accumulator B was set to PORTB such that it was visible on the LEDs.

The program was simulated using CodeWarrior and was seen to behave as expected. It was also loaded to the board and the LEDs were seen to change and count from 0 to A twice. When running the program on the Dragon board, the number of iterations performed by the delay loop was increased to slow the count down such that the LED changes were visible.

4.1 Code

```
CountUpSR:
        INCA
        CBA ; Compare A and B, is it time to reset?
        BNE DoneReset
        LDAA #$00 ; Reset
DoneReset:
        STAA PORTB ; Update output
        RTS
```

```
EntryA3:
        MOVB #$FF, DDRB
        MOVB #$FF, PORTB
        LDAA #$00
        LDAB #$0B ; When to reset A
        LDX #$0016 ; Howe many times to increment, 0xA*2

Repeat:
        JSR CountUpSR

        PSHA
        PSHX
        JSR EntryA2 ; Delay
        PULX
        PULA

        DEX
        BNE Repeat
        JMP HERE
```