ECE 372: Introduction To Microprocessors

Lab 2: Memory Dump: Loops and Memory Access

June 7th, 2018

Summer 2018

Honor Code: I have neither given nor received unauthorized assistance on this graded report.

x Brian Vera-Burgos x Srinivas Simhan

Table of Contents

Objective

- To become familiar with CodeWarrior
- To become familiar with basic loops and definitions in assembly language
- Calculate basic hexadecimal arithmetic by hand, and then follow up results using CodeWarrior

Equipment Used

- CodeWarrior

Prelab

The purpose of the prelab was to get us familiar with using functions print values to the terminal. The prelab had us accomplish this by printing the letters of the alphabet from A to Z.

Prelab Code

```
            lds   #ROMStart ; load stack pointer
            jsr TermInit
            LDAA #28 ; counter NOTICE; there is an oddity where if you cylce through the code to the end using f10 then this
;program works perfect,
 ;if you hit run then it will also print two junk values. if the counter is reduce to 26 then the opposite happens, two values are lost
;when doing f10
;and it works perfectly if the run button is used.
            LDAB #$41 ; "A" value
LOOP:
      jsr putchar  ;print out A
      INCB         ;Increment to the next letter
      DECA         ;decrement to the next
      CMPA #0      ;check if the counter has hit zero
      BNE LOOP
```

Procedure

In part A of the lab, we wrote a program that defined the following set of values at position 1000, and then printed them with a space in between each value:
$3A,$3B,$5F,$6F,$8E,$9D,$8B,$81,$1B,$1C,$1E,$16,$18,$19,$18,$18,$2A,$1,$3,$4

Explanation of Code

This was easily accomplished by first making a definition “Values” that stored each of the values above starting at $1000. Then we simply loaded X to be position $1000 and A to be a counter going to 20. In the Loop we load B to be the value at X’s position and print it. After this we print a space, increment X, Decrement A and compare A with 0. If A is greater than zero, run the program again.

In part B of the lab, we accomplished the same task with the added complexity of only printing a value out when the user hits the space bar. It was specifically added that no values should be printed if any key other than the spacebar is hit.

Explanation of Code

This program works the exact same way as in part A, but with an added loop that checks the users input and looks to see if that input equals the space bar in ascii code.

In part C of the lab we added to our program by adding a counter of how many numbers were printed out followed by a colon before printing the values.

Code for Part A

```
;-------------------------------------------------------
; Variable/Data Section
```

```
;---------------------------------------------------
            ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here

values dc.b $3A,$3B,$5F,$6F,$8E,$9D,$8B,$81,$1B,$1C,$1E,$16,$18,$19,$18,$18,$2A,$1,$3,$4 ;The values that we are going to
print out

        INCLUDE 'utilities.inc'
        INCLUDE 'LCD.inc'
;---------------------------------------------------
; Insert your code here
;---------------------------------------------------
        lds   #ROMStart ; load stack pointer
        jsr TermInit

        LDX #$1000 ;program runs counter
        LDAA #20   ;counter

LOOP
        LDAB 0,X    ;Load in the next value to B
        JSR out2hex ;Print out the next value

        LDAB #$20   ;Load a space
        JSR putchar ;print out a space

        INX         ;increment X to the next value
        DECA        ;decrement the nummber of times the program has run
        CMPA #0     ;check if the program has run 20 times and if so end mainloop
        BNE LOOP




here: jmp here          ;Stay here forever to end program
```

```
;-------End Program Lab#1----------
```

Code for Part B

```
;---------------------------------------------------------
; Variable/Data Section
;---------------------------------------------------------
            ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here

values dc.b $3A,$3B,$5F,$6F,$8E,$9D,$8B,$81,$1B,$1C,$1E,$16,$18,$19,$18,$18,$2A,$1,$3,$4 ;The values that we are going to
print out


        INCLUDE 'utilities.inc'
        INCLUDE 'LCD.inc'
;---------------------------------------------------------
; Insert your code here
;---------------------------------------------------------
        lds   #ROMStart ; load stack pointer
        jsr TermInit


        LDX #$1000 ;program runs counter
        LDAA #20   ;counter

LOOP
        LDAB 0,X    ;Load in the next value to B
        JSR out2hex ;Print out the next value

        LDAB #$20   ;Load a space
        JSR putchar ;print out a space

        INX         ;increment X to the next value
        DECA        ;decrement the nummber of times the program has run
        CMPA #0     ;check if the program has run 20 times and if so end mainloop
        BNE LOOP
```

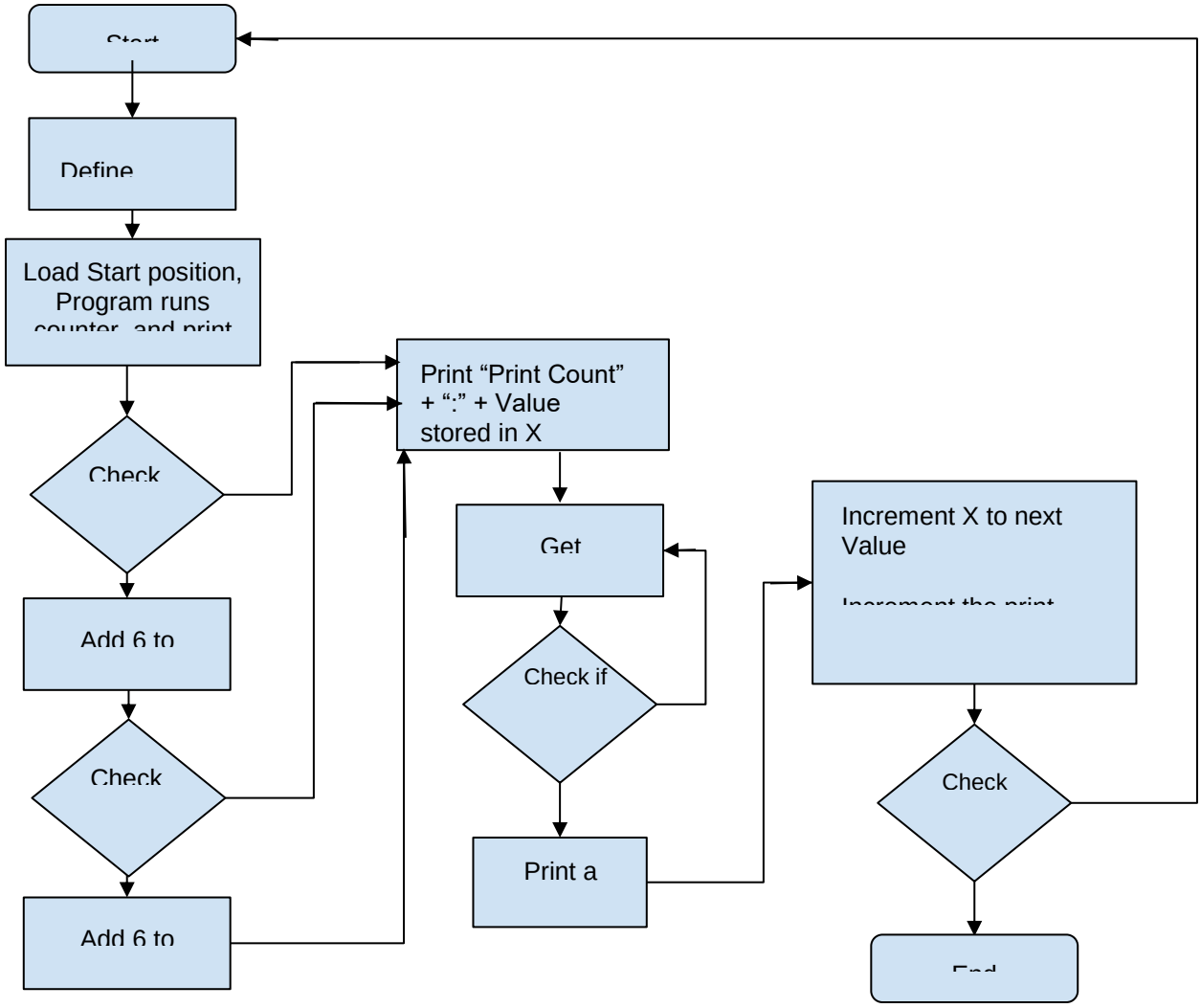
```
here: jmp here          ;Stay here forever to end program

;-------End Program Lab#1-------------
```

Code for Part C

```
;---------------------------------------------------------
; Variable/Data Section
;---------------------------------------------------------
            ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here

values dc.b $3A,$3B,$5F,$6F,$8E,$9D,$8B,$81,$1B,$1C,$1E,$16,$18,$19,$18,$18,$2A,$1,$3,$4 ;The values that we are going to
print out
printcount dc.b $01  ;Will be used to print which number the program has printed out

       INCLUDE 'utilities.inc'
       INCLUDE 'LCD.inc'
;---------------------------------------------------------
; Insert your code here
;---------------------------------------------------------
        lds   #ROMStart ; load stack pointer
        jsr TermInit

        LDX #$1000; starting position
        LDAA #20   ;program runs counter
        LDY #$1014 ;Print count position

MAINLOOP

COUNTLOOP
        LDAB 0,Y
        CMPB #$0A    ;Check to see if the counter is about to overflow above 10
        BLO CONTINUE ;if it isnt jump ahead to continue
        ADDB #$06    ;if it is add 6 to make the value $10 so it looks like decimal 10
        CMPB #$1A    ;Check to see if the counter is about to overflow above 20
```

```
        BLO CONTINUE ;if it isnt jump ahead to continue
        ADDB #$06    ;if it is add 6 to make the value $20 so it looks like decimal 20

CONTINUE

        JSR out2hex  ;print current counter value (essentially the same thing as accumulator A)
        LDAB #$3A    ;Load a :
        JSR putchar  ;Print out a :
        LDAB 0,X     ;Load in the next value to B
        JSR out2hex  ;Print out the next value

SPACELOOP
        JSR getchar  ;Get input from keyboard
        CMPB #$20    ;Check if it is the spacebar
        BNE SPACELOOP;if not go back to spaceloop and get another input

        LDAB #$20    ;Load a space
        JSR putchar  ;print out a space

        INX          ;increment X to the next value
        INC printcount    ;increment printcount
        DECA         ;decrement the nummber of times the program has run
        CMPA #0      ;check if the program has run 20 times and if so end mainloop

        BNE MAINLOOP

here: jmp here         ;Stay here forever to end program

;-------End Program Lab#1-------------
```

Flowchart (Part C)



Post Lab Questions

1) Briefly explain what the overall purpose of this program is and complete the memory table below which should show what is in memory after the program has been run.

```
        ORG   $1000
TABLE   DC.B  $18, $39, $E1, $8B, $15

        ORG   $2000
        LDX   #TABLE
        LDAB  #5
LOOP    LDAA  0, X
        STAA  5, X
        INX
        DBNE  B, LOOP
        BRA   *
```

First it should be noted that the LDX #TABLE command does not load anything into X. If this fact is ignored, this program is intended to take the 5 values in TABLE and copy them 5 positions higher in the memory. The resulting memory after running this program would be as follows would be as follows:

$2000	$18
$2001	$39
$2002	$E1
$2003	$8B
$2004	$15
$2005	$18
$2006	$39
$2007	$E1
$2008	$8B
$2009	$15

2) Assuming you have the same data section as in the above, write a program that adds 1 to each of the five numbers in memory. This must be done using a program loop.

The previous question can be completed with minimal modifications to the given program:

```
;---------------------------------------------------
; Variable/Data Section
;---------------------------------------------------
          ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here
    ORG $1000
TABLE dc.b $18,$39,$E1,$8B,$15
      INCLUDE 'utilities.inc'
      INCLUDE 'LCD.inc'
;---------------------------------------------------
; Insert your code here
;---------------------------------------------------
       lds   #ROMStart ; load stack pointer

       ORG $2000
       LDX #TABLE
       LDAB #5
   LOOP
       LDAA 0,X   ;Get the value of X
       INCA       ;add one to the value
       STAA 0,X   ;Put the value back
       INX        ;Go to the next value
       DBNE B,LOOP;decrement the count value
       BRA *
here: jmp here         ;Stay here forever to end program

;-------End Program Lab#1-------------
```

3) What is the difference between ”For”, ”While” and ”Do While” loop?

The difference between a for loop and a while loop is that a for loop allows you to define its looping conditions as well as conditions for incrementing in the same line that you define the loop. Meanwhile a while loop only allows you to define the conditions that break the loop in its definition line. The difference between a while loop and a do-while loop is that a do-while loop is always run at least

once. More specifically, a while loop will check its break conditions on the first time through while a do-while loop will not check until the second time through the loop.

Conclusion

In the prelab we simply learned how to use various programs to print values to the terminal. In part A we learned how to make a definition to store a list of values in the memory. We further used the print commands from the prelab to print out these values in succession.

In part B we took this a step further by learning how to make a loop that would take input from the keyboard, and using commands from the previous lab create a simple algorithm to determine if the key pressed was the spacebar. The program would then wait until it detected the spacebar to print the next value.

In part C we finished by adding a printed counter that would print the current count of the program. This part turned out to be more complicated then at first glance, once the count reached 10 (in decimal) it would go to 1A (in hex). In order to solve this and print 10 instead of 1A we learned to create a branching if statement that would check if the count value was equal to or greater than 10 and then to add 6 to it. This would create a hex number that appears like the decimal number we are looking for.