Royal Military College of Canada

Department of Electrical and Computer Engineering

Lab report for the course

EEE351

Lab # 05

Bubble Sort

SUBMITTED TO

Dr. CHABINI

BY

OCdt Brown and OCdt Gillingham

DATE : 2 November 2022

1. Introduction and problem statement

Lab 5 was designed in order to further cement our understanding of assembly code development for simple tasks with an emphasis on various addressing modes covered in class. The goal for this lab was to write an algorithm with subroutines to sort a series of given numbers stored in memory from low to high values. The values would then be stored as 8-bit values to practice using numerical data. Some constraints we were required to abide by consist of using at least one subroutine, accounting for a zero-size list (assuming the null-delimiter remains), and using the bubble sort algorithm.

2. Problem solution

The first step was to copy all numbers from ROM into RAM. After all numbers are placed in RAM, the bubble sort algorithm can be run. In order to create a bubble sort algorithm a loop with a nested for loop was required. The master loop is repeated if a swap has occurred. The nested for loop will repeat if the current number is not the null char of $FF. Within the nested for loop accumulator A and B are loaded with the memory location pointed to by register Y and Y + 1 respectively. If the value of B is less than the value in A the two values' memory location is swapped using the swap subroutine. If the value of B is greater than A the for loop is repeated. After the for loop it is checked if a swap bit occurred by referencing the swap boolean bit. If a swap has occurred the loop is repeated otherwise the list has been sorted and the program is exited.

3. Testing results

As seen in *Appendix: figure one* through three, we used three separate datasets to conclude our bubble sort algorithm worked successfully. The first of the three lists given to us in the lab handout by the professor consisted of 24 numbers base 10. As seen in figure four the list was successfully sorted and placed in RAM. After sorting the first set, we created a set of 24 numbers base 16, which ran with no problem shown in figure five. Finally, we made and tested an empty set that included the null delimiter. This did nothing, as seen in figure six.

4. Answers to questions

N/A

5. Difficulties faced

Several difficulties were faced when trying to compare numbers, especially with the null delimiter FF. When trying to compare FF, 1111 1111 in binary, if using a signed command like BGT it will be considered as a negative number, which is not the case in our situation. This is why we had to use an unsigned command like BHT to get the job done.

6. Conclusions

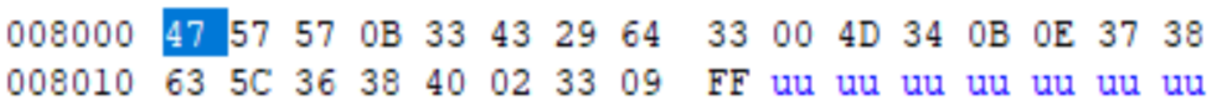
In conclusion, this lab has deepened our understanding of branching and basic assembly operations. In this lab we successfully created a bubble sort algorithm using the assembly language for the M68HC12 microprocessor.

7. References

Bubble sort: https://en.wikipedia.org/wiki/Bubble_sort

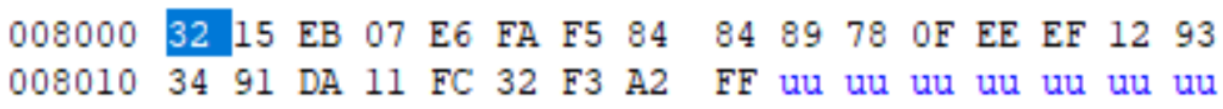
8. Appendix

Figure 1: NewList unsorted in ROM



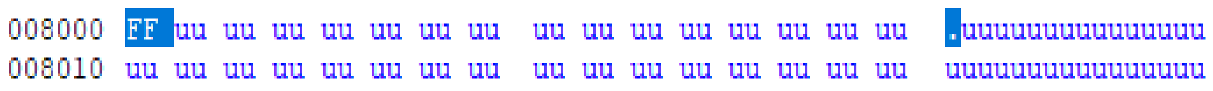
```
008000  47 57 57 0B 33 43 29 64   33 00 4D 34 0B 0E 37 38
008010  63 5C 36 38 40 02 33 09   FF uu uu uu uu uu uu uu
```

Figure 2: HexList unsorted in ROM



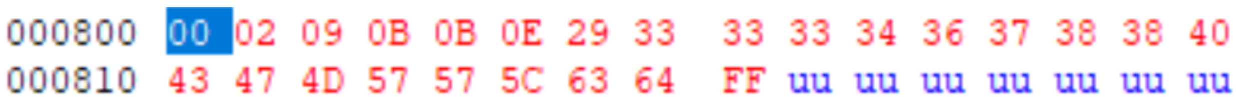
```
008000  32 15 EB 07 E6 FA F5 84   84 89 78 0F EE EF 12 93
008010  34 91 DA 11 FC 32 F3 A2   FF uu uu uu uu uu uu uu
```

Figure 3: Empty unsorted in ROM



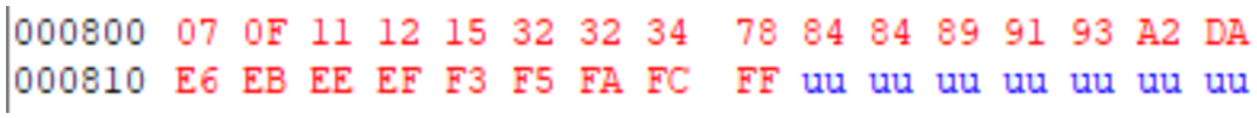
```
008000  FF uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   .uuuuuuuuuuuuuuu
008010  uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
```

Figure 4: NewList sorted in RAM



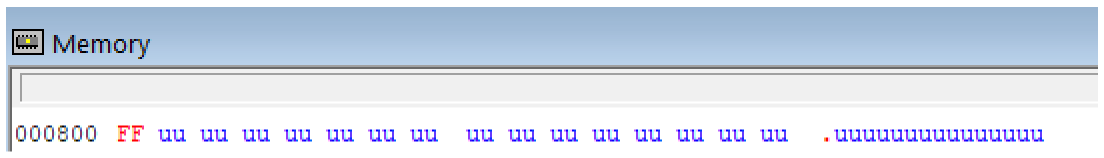
```
000800  00 02 09 0B 0B 0E 29 33   33 33 34 36 37 38 38 40
000810  43 47 4D 57 57 5C 63 64   FF uu uu uu uu uu uu uu
```

Figure 5: HexList sorted in RAM



```
000800  07 0F 11 12 15 32 32 34   78 84 84 89 91 93 A2 DA
000810  E6 EB EE EF F3 F5 FA FC   FF uu uu uu uu uu uu uu
```

Figure 6: Empty sorted in RAM



```
000800  FF uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   .uuuuuuuuuuuuuuu
```

Figure 7: Code

```
;**************************************************************************
;*
;* File: main.asm
;*
;* Author: OCdt Brown & OCdt Gillingham
;*
;* Description: This is a file for sorting an array of integers. The integers
;*              are already placed into ROM. It will check for an array terminator
;*              symbol of $FF. If not at end of array will call swap. By looping
;*              through this process untill no swaps occure a bubble sort algo
;*              will be compleated.
;*
;**************************************************************************

; export symbols
            XDEF Entry              ; export 'Entry' symbol
            ABSENTRY Entry          ; for absolute assembly: mark this as application entry point



; Include derivative-specific definitions
    INCLUDE 'derivative.inc'

StartRAM    EQU  $0800  ; where sorted list will be stored
StartROM    EQU  $8000  ; where unsorted list will be stored
Program     EQU  $C000  ; where program will be stored
Swapped     EQU  $BFFF  ; will store if a swap occured
                         ; did I spell occured correctly

; variable/data section
            ORG Swapped  ; stres if a swap occured
            DC.B 00      ; 01 is true

            ORG StartROM ;where the unsorted list is stored

            ;thise are dec numbers
;NewList      DC.B 71, 87, 87, 11, 51, 67, 41, 100
;           DC.B 51, 0 , 77, 52, 11, 14, 55, 56
;           DC.B 99, 92, 54, 56, 64, 2, 51, 9
;           DC.B $FF

;second test
;HexList     DC.B $32, $15, $EB, $07, $E6, $FA, $F5, $84
;            DC.B $84, $89, $78, $0F, $EE, $EF, $12, $93
;            DC.B $34, $91, $DA, $11, $FC, $32, $F3, $A2
;            DC.B $FF

Empty       DC.B $FF
```

```
; code section
            ORG   Program

Entry:
            ; start up stuff
            ldx  #StartROM
            ldy  #StartRAM

            ; loads all the numbers into RAM
startloop   ldaa 1,x+
            staa 1,y+
            cmpa #$FF
            bne  startloop

            ;loop is used to sort
loop        ldy  #StartRAM

            ; for each pair of bits check if in right order
for         ldaa 1,y+
            ldab 0,y

            ;if at end of list end for loop
            cmpb #$FF
            beq  skip

            ;if b is greater than a branch to swap
            cba
            bhi  swap

            ;run for again
afterswap   bra  for

            ;after null terminator is reached store null and
            ;check if a swap occured, if so loop again
skip        ldaa Swapped  ;temp hold of swapped bit
            cmpa #01
            ; set swap bit to false
            ldaa #00
            staa Swapped
            beq  loop
```

```
;***************************************************************************
;*
;* Subroutine: Decrypt
;*
;* Description: This subroutine is where the algo will swap the order the numbers
;*              are stored in memory.
;*
;* Inputs: A -  contains the value that needs to be stored second
;*         B -  contains the value that needs to be stored first
;*         IY - contains the address to store the values in A and B
;*
;* Outputs: None
;*
;* Subroutines: None
;***************************************************************************
swap:
            dey
            stab 1,y+
            staa 0,y
            ldaa #01
            staa Swapped
            bsr  afterswap
            rts


;**************************************************************
;*                Interrupt Vectors                           *
;**************************************************************
            ORG   $FFFE
            DC.W  Entry           ; Reset Vector
```