ECE3120: Computer Systems
Chapter 4: Subroutines

Manjeera Jeedigunta
http://blogs.cae.tntech.edu/msjeedigun21
Email: msjeedigun21@tntech.edu
Tel: 931-372-6181, Prescott Hall 120
Stack Frame

- The region in the stack that holds incoming parameters, the subroutine return address, local variables, and saved registers is referred to as stack frame.
- The stack frame is also called **activation record**.

![Figure 4.9 Structure of the 68HC12 stack frame](image-url)
Example: Convert the character and word counting program into a subroutine. The starting address of the string is passed to this subroutine in the index register X, and the character and word count are returned to the caller in the register D and X respectively.

```
tab        equ        $09
sp         equ        $20
cr          equ        $0D
lf          equ        $0A
ch_cnt    equ        0 ;offset
wd_cnt    equ        1 ;offset
string_x  fcc        "this is a long and strange test string to count chars and words."
            fcb        0
            org        $1000
char_cnt rmb        2
word_cnt rmb        2
            org        $1500
ldx        #string_x ;base address of the string
jsr        sub_cwcnt
std        char_cnt
stx        word_cnt
swi

; subroutine for character and word count

sub_cwcnt leas        -2,sp ;allocation for the local variables
cr          ch_cnt,sp
cr          wd_cnt,sp
string_lp  ldab        1,x+ ; get one character and move string pointer
           lbeq        done ; is this the end of the string?
           inc        ch_cnt,sp
```
; the following 8 instructions skip white space characters between words
    cmpb #sp
    beq string_ip ; if it is any of these it just moves to the next
    cmpb #tab
    beq string_ip
    cmpb #cr
    beq string_ip
    cmpb #lf
    beq string_ip

; a non-white character is the start of a new word
    inc wd_cnt,sp

wd_loop    ldab 1,x+ ; get one character and move pointer
    beq done
    inc ch_cnt,sp

; the following 8 instructions check the end of a word
    cmpb #sp
    lbeq string_ip
    cmpb #tab
    lbeq string_ip
    cmpb #cr
    lbeq string_ip
    cmpb #lf
    lbeq string_ip
    bra wd_loop

done       ldab wd_cnt,sp
    clra
    xgdx
    ldab ch_cnt,sp
    clra
    leas 2,sp
    rts

end
Data Conversion: Lower to Upper case

- Lower case letters range from 97-122 (decimal).

- Upper case letters range from 65-90 (decimal).

- Notice that the lower case letter values are exactly 32 greater than their upper case counterparts. This is useful for upper and lower case conversions.

- The numbers 0-9 range from 48-57 (decimal).
Write a subroutine to convert the lower case letters in a string into uppercase letters. The string address is passed to the subroutine through the index register X

```
org $1000
string fcc "This is a String to be converted." ;string to be converted
db 0
org $1500
ldx #string
jsr locase2hi
swi

locase2hi psha
loopl2h ldaa 0,x
beq done
cmpa #$61 ; decimal 97
blo next
cmpa #$7A ;decimal 122
bhi next
suba #$20 ;decimal 32
staa 0,x
next inx
bra loopl2h
done pula
rts
end
```

Lower case – 97-122
Upper case -65 -90
Numbers – 48-57
Lower>Upper by 32
Bubble Sort

- Sorting is useful for improving the searching speed when an array or a file need to be searched many times.
- Bubble sort is a simple but inefficient sorting method.

0 157 13 35 9 98 810 120 54 10 30

1 13 35 9 98 157 120 54 10 30 810

2 13 9 35 98 120 54 10 30 157 810

3 9 13 35 98 54 10 30 120 157 810

4 9 13 35 54 10 30 98 120 157 810

5 9 13 35 10 30 54 98 120 157 810

6 9 13 10 30 35 54 98 120 157 810

7 9 10 13 30 35 54 98 120 157 810 sorted

8 9 10 13 30 35 54 98 120 157 810

9 9 10 13 30 35 54 98 120 157 810

- All elements in positions > or equal to n-i are in right position after iteration I

Hence

1st iteration = n-1 comparisons

2nd iteration = n-2 comparisons so on

• N-1 iterations might not be required to sort the array

An array is sorted if no swaps are made in an iteration
Example 4.13: Write a subroutine to implement the bubble sort algorithm and a sequence of instructions along with a set of test data for testing this subroutine.

- Pass the base address of the array and the array count in the stack
- Four bytes are needed for local variables.

Figure 4.15 Stack frame for bubble sort
Inner = no. of swaps

Not in order so set the flag to 0

Move to next iteration

Figure 4.14 Logic flow of bubble sort
arr equ 13 ; distance of the variable arr from stack top
arcnt equ 12 ; distance of the variable arcnt from stack top
buf equ 3 ; distance of local variable buf from stack top
in_order equ 2 ; distance of local variable in_order from stack top
inner equ 1 ; distance of local variable inner from stack top
iteration equ 0 ; distance of local variable iteration from stack top
true equ 1 ; used to set in_order flag
false equ 0
n equ 30 ; array count
local equ 4 ; number of bytes used by local variables
org $800
array_x db 3, 29, 10, 98, 54, 9, 100, 104, 200, 92, 87, 48, 27, 22, 71
        db 1, 62, 67, 83, 89, 101, 190, 187, 167, 134, 121, 20, 31, 34, 54
org $1000
lds #$8000 ; initialize stack pointer
ldx #array_x
pshx
ldaa #n
psha
jsr bubble
leas 3, sp ; deallocate space used by outgoing parameters
swi

Figure 4.15 Stack frame for bubble sort
bubble  pshd

  pshy
  pshx
  leas  -local,sp ; allocate space for local variables
  ldaa  arcnt,sp ; compute the number of iterations to be performed
defa   ;A ← N-1
  staa  iteration,sp ;Iteration←N-1

ploop  
  ldaa  #true ; set array in_order flag to true before any iteration
  staa  in_order,sp ;in_order←1
  ldx   arr,sp ; use index register X as the array pointer
  ldaa  iteration,sp ; initialize inner loop count for each iteration
  staa  inner,sp ; inner←N-1 (1st iter)

clloop
  ldaa  0,x ; compare two adjacent elements
  cmpa  1,x ; "
  bls   looptest

; the following five instructions swap the two adjacent elements
  staa  buf,sp ; swap two adjacent elements
  ldaa  1,x ; "
  staa  0,x ; "
  ldaa  buf,sp ; "
  staa  1,x ; "
  ldaa  #false ; reset the in-order flag
  staa  in_order,sp ; "

Figure 4.15 Stack frame for bubble sort
looptest

inx

dec inner,sp
bne cloop
tst in_order,sp ; test array in_order flag after each iteration
bne done
dec iteration,sp
bne ploop
done
leas local,sp ; deallocate local variables
pulp
pulp
puld
rts
end

Figure 4.15 Stack frame for bubble sort
Next

- Using the D-Bug 12 Functions to perform I/O Operations