Save as Permanent

Save as Fillable

MIE438 – Microprocessors and Embedded Microcontrollers

Laboratory Experiment 4

Deliverable

Spring 2018

Department of Mechanical and Industrial Engineering

University of Toronto

Notes:

Practical Section: PRA 0102

Last Name, First Name	Student Number
Moolayil, Sidhanth	1004101679

1. Using Lab4_2.mcp, write out the numerical values of the stack contents at location marked in the given code (this is the same location shown in the introduction). Include addresses for each value as well. What is the value of SP?

ADDRESS	CONTENT	TITLE/COMMENT
3FF5	00	Y High Byte
3FF6	00	Y Low Byte
3FF7	00	X High Byte
3FF8	00	X Low Byte
3FF9	2C	D High Byte
3FFA	39	D Low Byte
3FFB	40	PC High Byte
3FFC	37	PC Low Byte
3FFD	32	Return Value
3FFE	2C	Parameter 2
3FFF	16	Parameter 1

Value of Stack Pointer:

3FF5

2. Write and test code to call the subroutine a second time (immediately after the "PULA / INS / INS" code in MAIN). For parameter one, use the **returned value** from the first call to SUBR. For parameter two, use a constant value of your choice. Write your code here:

```
LDAA    2,SP; first parameter = data at [SP+2]
PSHA        ; push accumulator A
LDAA    #50 ; arbitrary value
PSHA
DES         ; blanck space for return val
JSR         ; jump to subroutine SUBR
```

3. Record your calling code from MAIN and from your subroutine. You may attach a printout instead.

```
main:
            LDAA    #$FF
            STAA    DDRJ
            STAA    DDRB
            STAA    DDRP
            STAA    DDRA
            STAA    DDRK
            LDAA    #$00
            STAA    DDRH
            LDAA    #$0F
            STAA    PTP
            CLR     PTJ
            CLR     PORTB


            JSR     LONGWAIT
            JSR     INIT_LCD

            ; Call the routine to print first name

            LDX     #STRING1
            LDAB    #0
            JSR     PRINT_STRING

            LDAA    #$C0
            JSR     COMWRT4

            ; Call the routine to print last name

            LDX     #STRING2
            JSR     PRINT_STRING

            JMP     BACK

PRINT_STRING:

            ;Printing subroutine, input string address on X => output print string on LCD

            LDAA    B,X
            JSR     DATWRT4
            ADDB    #1
            LDAA    B,X
            CMPA    #36
            BNE     PRINT_STRING
            LDAB    #0
            RTS
```

4. Would you consider your subroutine safe? Is there any input or use that might cause unexpected behavior?

Utilizing the stack to retreive the address the code has to start at for return and jump functions can be unsafe if the stack has been updated (computer will start at and run other, unintended code resulting in unexpected behaviour).

5. How is the parameter for the string passed to the writeLCD function? How are the other local variables (such as 'int i' in writeLCD) stored?

String arguement is loaded into accumulator D (both A and B) before jumping to the writeLCD subroutine (function) and accessed. int i is stored in the stack.

6. Now, how is the parameter passed to the writeLCD function? How are the other local variables (such as 'int i') stored? Were there any changes? Speculate on why or why not.

String arguement is loaded into register X and accumulator B before CALLing the writeLCD subroutine (function). int i is again stored in the stack. The change from the previous implementation was instead of storing the string into the 16 bit width accumulator D register, it was stored in the general purpose register X and accumulator B register. It is speculated that registers X and Y are more efficient for addressing/ 'memory usage' than accumulators, based on the observed changes in the code.

7. Of the two compiled versions of the code, was one (significantly) shorter? Which do you think will be faster, and why?

The two compiled versions of the code were approximately the same length. In terms of speed, since length is almost the same, the only factor that would affect this would be the use of accumulator D vs accumulator B and general purpose register X. Since accumulators are faster than general purpose registers, it can be expected that the small memory mode (solely using accumulator D) would run faster.