

LAB 4 REPORT

1. THEORY TOPICS

- **Assembler directives:** Assembler directives tell the assembler to set the data at particular addresses, allocate space in memory for constants and variables, define synonyms or include additional files. The main directives I have used in this lab are the `.data` directive, which places variables under this section in the RAM memory, and some constant initialization directives like `.int` (integer) or `.cstring` (similar to `.string` but `.cstring` adds a null character at the end of the array of characters so it's easier to know when the array ends).

Where you place your variables is important depending on the task you need to do. If you place variables in the `.text` section these will be placed in flash memory and will only be available to read operations. However the `.data` section places information in the RAM memory which is read and write.

- **Addressing modes:** Addressing modes are used to tell the compiler how you are accessing a register or a memory location. There are a total of 7 addressing modes: *register*, *indexed*, *symbolic*, *absolute*, *indirect register*, *indirect autoincrement*, and *immediate*.

Indirect autoincrement is an addressing mode only valid for source operands and with syntax `@Rn+`. The effective address is the content of the address pointed by the register. After this, the register is incremented by +1 if the operation is byte-size and by 2 if the operation is word-size.

For example if R6 is pointing to address 0x2402 which holds the string "oh" and we have `mov.b @R6+, R7`, the instruction would look like this:

$R7 \leftarrow M[R6]$; $R6 \leftarrow R6 + 1$. So R7 will end up having the character 'o' and R6 will now point to address 0x2403 where 'h' resides.

2. PROGRAM 1

The screenshot displays an IDE with three main panels:

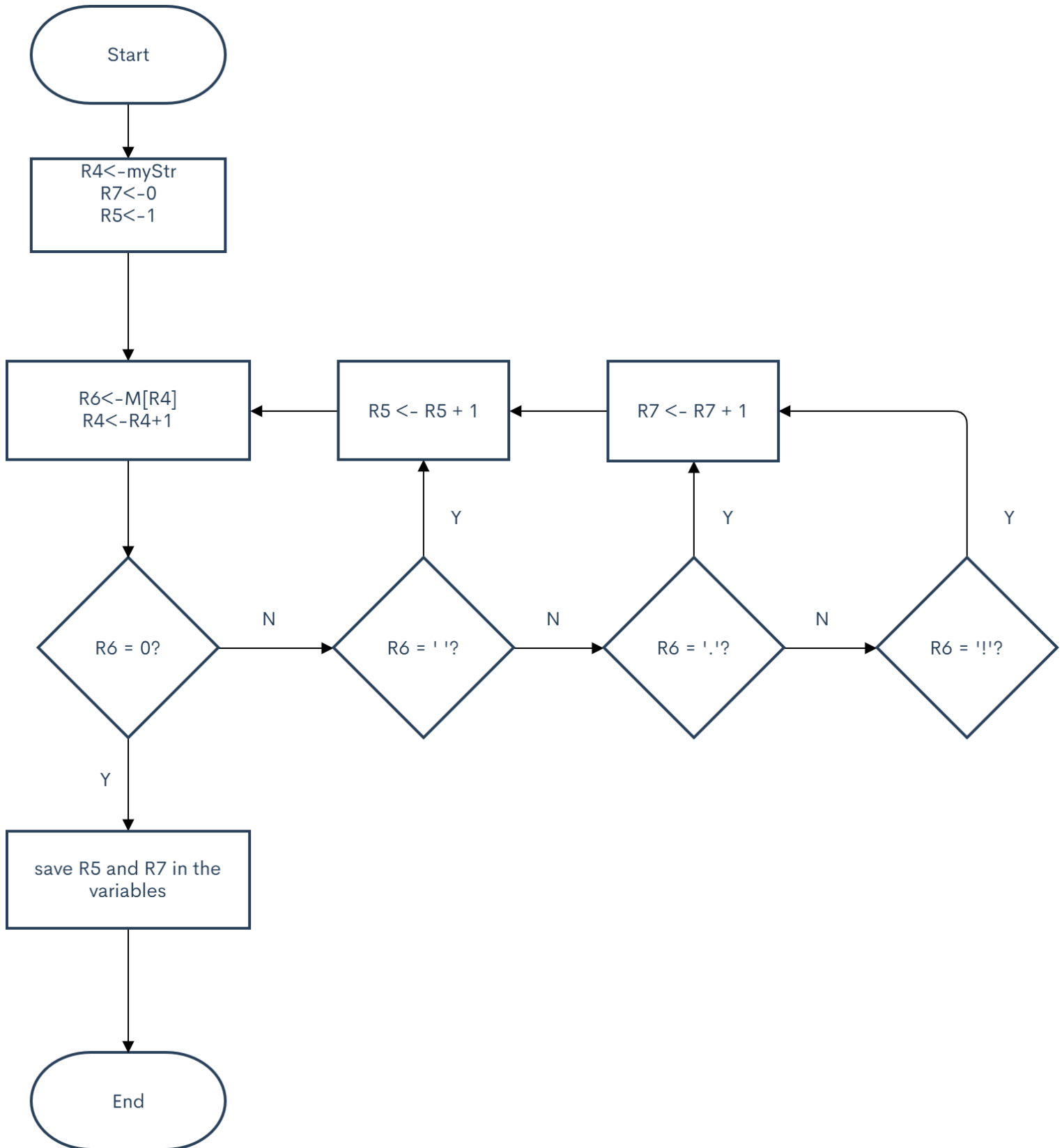
- Left Panel (Registers):** A table of registers and their values.

Name	Value
PC	0x004460
SP	0x004400
SR	0x00F3
R3	0x000000
R4	0x00441E
R5	0x000006
R6	0x000000
R7	0x000002
R8	0x00A508
R9	0x000000
R10	0x0FFFFF
R11	0x0A2E38
R12	0x000000
R13	0x000002
R14	0x000182
R15	0x00000B
- Center Panel (Assembly Code):** Shows assembly code for a program.


```

29 .retain ; Override ELF conditional linking
30 ; and retain current section.
31 .retainrefs ; And retain any sections that have
32 ; references to current section.
33
34 ;-----
35 RESET: mov.w #_STACK_END,SP ; Initialize stack pointer
36 mov.w #WDTPW|WDTHOLD,&WDCTL ; Stop watchdog timer
37
38 ;-----
39 ; Main loop here sentence: "HELLO WORLD. I AM THE MSP430!"
40 ;-----
41 main:
42 mov.w #myStr, R4 ; load the starting address of the string into
43 clr.b R7 ; register R7 will store the sentence count
44 mov.b #1, R5 ; R5 = 1 because word count = number of ' ' +
45 gnext: mov.b @R4+, R6 ; get a new character
46 cmp #0, R6 ; is it a null character
47 jeq lend ; if yes, go to the end
48 cmp.b #' ', R6 ; is it an ' ' character
49 jne dot ; if not, go to dot
50 inc.w R5 ; if yes, increment word counter
51 jmp gnext ; next character
52
53 dot: cmp #' ', R6 ; is it a ' '
54 jne exc ; if not go to exc
55 inc.w R7 ; if yes increment sentence counter
56 jmp gnext ; next character
57
58 exc: cmp #'!', R6 ; is it a '!'
59 jne gnext ; if not next character
60 inc.w R7 ; if yes increment sentence counter
61 jmp gnext ; next character
62
63 lend: mov.w R5, &word_count ; write result in R7 and r5
64 mov.w R7, &sent_count
65 bis.w #LPM4, SR ; LPM4
66 nop ; required only for Debugger
67
68
69 ;-----
70 ; Stack Pointer definition
71
      
```
- Right Panel (Memory Dump):** Shows a memory dump starting at address 0x2402.

Address	Hex	ASCII
0x0023EA	USB_Operation_USBIEPBCTX_5	
0x0023EB	F3FC C4DD	
0x0023EC	USB_Operation_USBIEPBCTX_5	
0x0023ED	F687	
0x0023EE	USB_Operation_USBIEPCNF_6	
0x0023EF	D3B9	
0x0023F0	USB_Operation_USBIEPBCTX_6	
0x0023F1	B6DE DDEE	
0x0023F2	USB_Operation_USBIEPBCTX_6	
0x0023F3	AD76	
0x0023F4	USB_Operation_USBIEPCNF_7	
0x0023F5	5E9E	
0x0023F6	USB_Operation_USBIEPBCTX_7	
0x0023F7	FF73 C50F	
0x0023F8	USB_Operation_USBIEPBCTX_7	
0x0023F9	54EB 0002 0000 DD1F A7C6 FB87 531B	
0x0023FA	ACB8 B87E EFCB 28FD ECD7 F4FB F4FA	
0x0023FB	FFFF D7FD 687E 7CDD CEF2 2E3F E84D	
0x0023FC	5ED7 266F F77E FD37 D2F7 85AB DF76	
0x0023FD	F368 9852 7EFA 26FF ACFB 1DFF 64C5	
0x0023FE	FEF4 8EF5 3FB6 DE76 2BE3 7DA5 6AFE	
0x0023FF	CDD0 FE75 F2E7 CDA7 B4F1 D55A E5BA	
0x002400	FFF8 45F7 736D 68EA CF45 8997 731F	
0x002401	BE9F EAE7 F766 7FF8 21CD E9B4 EFD0	
0x002402	4D5F FFEF F6AF C3AD FC1E 8ADF B9DF	
0x002403	673F 03FD B759 A751 0FDF FFFB 93A8	
0x002404	63F7 BAF6 DE96 D757 FEEE B45F EED6	
0x002405	F369 CF74 37D5 F8FB FFA0 4EE4 F775	
0x002406	22FE E928 7987 37F7 9C5E 9397 6798	
0x002407	73FB 8F3C 3665 66EE 971F F7BF B987	
0x002408	9397 9AD5 ECC5 675E BAFD CECB 9FDF	
0x002409	47FA B1DF 205F 0899 3FDA 7FDB 1D5A	
0x00240A	BEE3 D7DF EA97 EA5D 73D5 7078 CEED	
0x00240B	BF45 F73A CD22 4F72 F7E4 88A6 53F7	
0x00240C	ECAA EFAF 0CD9 3BE3 E791 FF0B 74F9	
0x00240D	3BD7 3AFA AE7E FBAE F0D4 BC76 3CDA	
0x00240E	C7D5 DFF6 A6DB 63E8 EF91 CAF6 EAFD	
0x00240F	F757 B6FB 9ED8 D96D 5F5F 5EDF C77E	



3. Program 2

Name	Value	Description
> DMA		
> Flash		
> MPY_16_Multiplier_16_Bit_Mode		
> MPY_32_Multiplier_32_Bit_Mode		
> Port_A		
> Port_1_2		
> P1IN	0xFC	Port 1 Input Pins
> P1OUT	0x00	Port 1 Output Pins
> P1DIR	0x00	Port 1 Direction Pins
> P1REN	0x00	Port 1 Enable Pins
> P1DS	0x00	Port 1 Drive Pins
> P1SEL	0x00	Port 1 Select Pins
> P1IV	0x0000	Port 1 Interrupt Vector
> P1IES	0x00	Port 1 Interrupt Enable Select
> P1IE	0x00	Port 1 Interrupt Enable
> P1IFG	0x00	Port 1 Interrupt Flag
> P2IN	0xFD	Port 2 Input Pins
> P2OUT	0x06	Port 2 Output Pins
> P2DIR	0x00	Port 2 Direction Pins
> P2REN	0x00	Port 2 Enable Pins
> P2DS	0x00	Port 2 Drive Pins
> P2SEL	0x00	Port 2 Select Pins
> P2IV	0x0000	Port 2 Interrupt Vector
> P2IES	0x00	Port 2 Interrupt Enable Select
> P2IE	0x00	Port 2 Interrupt Enable
> P2IFG	0x00	Port 2 Interrupt Flag
> Port_B		

```

25
26 ;-----
27 ; Main loop here doing: 4-3+5
28 ;-----
29 main:  mov.w  #operation, R4
30
31 next:  mov.b  @R4+, R6      ; next character
32        cmp.b  #0, R6      ; is it a null char?
33        jeq    lend        ; if yes go to end
34        cmp.b  #'+', R6    ; is it a +?
35        jne    sym         ; if not go to sym
36        jmp    sum         ; if yes go to sum
37
38 sym:   cmp.b  #'-', R6    ; is it a '-'
39        jne    num         ; if not go to num
40        jmp    subs        ; go to subs
41
42 sum:   mov.b  @R4+, R6    ; read next character
43        sub    #48, R6     ; subtract 48 to get number and put it in R6
44        add    R6, R7      ; add R6 and R7
45        jmp    next        ; go to next
46
47 num:   sub    #48, R6     ; subtract 48 to get number and put it in R6
48        mov.b  R6, R7      ; move contents of R6 into R7
49        jmp    next        ; go to next
50
51 subs:  mov.b  @R4+, R6    ; read next character
52        sub    #48, R6     ; subtract 48 to get number and put it in R6
53        sub    R6, R7      ; do R7<-R7 - R6
54        jmp    next        ; go to next
55
56 lend:  mov.b  R7, &P2OUT ; put R7 into P2OUT
57        bis.w  #LPM4, SR   ; LPM4
58        nop              ; required only for Debugger
59
60

```

4. Bonus

```

14 ;-----
15 .text                ; Assemble into program memory.
16 .retain              ; Override ELF conditional linking
17                      ; and retain current section.
18 .retainrefs          ; And retain any sections that have
19                      ; references to current section.
20 ;-----
21
22 RESET  mov.w  #_STACK_END, SP      ; Initialize stackpointer
23 StopWDT mov.w  #WDTCTL, WDTCTL     ; Stop watchdog timer
24
25
26 ;-----
27 ; Main loop here sent = I enjoy learning msp430
28 ;-----
29 main:  mov.w  #sent, R4            ; load the starting address of the string into R4
30
31 next:  mov.b  @R4+, R5            ; next char
32        cmp.b  #0, R5            ; is it null?
33        jeq    end                ; if yes go to end
34        cmp.b  #97, R5           ; is it a lower case?
35        jge    upper             ; if it is (>=97) go to upper
36        jmp    next              ; if not read next
37
38 upper:  sub.b  #32, R5            ; get upper case
39        mov.b  R5, -1(R4)         ; put it in the location where the lower case was (before autoincrement)
40        jmp    next              ; go to next
41
42 end:    bis.w  #LPM4, SR          ; LPM4
43        nop                    ; required only for Debugger
44 ;-----
45 ; Stack Pointer definition
46 ;-----
47 .global __STACK_END
48 .sect .stack
49
50 ;-----
51 ; Interrupt Vectors
52 ;-----
53 .sect ".reset"          ; MSP430 RESET Vector
54 .short RESET
55
56

```

Disassembly Memory Browser

0x2400

0x2400 <Memory Rendering 6>

Character	0x002400	0x002401	0x002402	0x002403	0x002404	0x002405	0x002406	0x002407	0x002408	0x002409	0x00240A	0x00240B	0x00240C	0x00240D	0x00240E	0x00240F	0x002410	0x002411	0x002412	0x002413	0x002414	0x002415	0x002416	0x002417	0x002418	0x002419	0x00241A	0x00241B	0x00241C	0x00241D	0x00241E	0x00241F	0x002420	0x002421	0x002422	0x002423	0x002424	0x002425	0x002426	0x002427	0x002428	0x002429	0x00242A	0x00242B	0x00242C	0x00242D	0x00242E	0x00242F	0x002430	0x002431	0x002432	0x002433	0x002434	0x002435	0x002436	0x002437	0x002438	0x002439	0x00243A	0x00243B	0x00243C	0x00243D	0x00243E	0x00243F	0x002440	0x002441	0x002442	0x002443	0x002444	0x002445	0x002446	0x002447	0x002448	0x002449	0x00244A	0x00244B	0x00244C	0x00244D	0x00244E	0x00244F	0x002450	0x002451	0x002452	0x002453	0x002454	0x002455	0x002456	0x002457	0x002458	0x002459	0x00245A	0x00245B	0x00245C	0x00245D	0x00245E	0x00245F	0x002460	0x002461	0x002462	0x002463	0x002464	0x002465	0x002466	0x002467	0x002468	0x002469	0x00246A	0x00246B	0x00246C	0x00246D	0x00246E	0x00246F	0x002470	0x002471	0x002472	0x002473	0x002474	0x002475	0x002476	0x002477	0x002478	0x002479	0x00247A	0x00247B	0x00247C	0x00247D	0x00247E	0x00247F	0x002480	0x002481	0x002482	0x002483	0x002484	0x002485	0x002486	0x002487	0x002488	0x002489	0x00248A	0x00248B	0x00248C	0x00248D	0x00248E	0x00248F	0x002490	0x002491	0x002492	0x002493	0x002494	0x002495	0x002496	0x002497	0x002498	0x002499	0x00249A	0x00249B	0x00249C	0x00249D	0x00249E	0x00249F	0x0024A0	0x0024A1	0x0024A2	0x0024A3	0x0024A4	0x0024A5	0x0024A6	0x0024A7	0x0024A8	0x0024A9	0x0024AA	0x0024AB	0x0024AC	0x0024AD	0x0024AE	0x0024AF	0x0024B0	0x0024B1	0x0024B2	0x0024B3	0x0024B4	0x0024B5	0x0024B6	0x0024B7	0x0024B8	0x0024B9	0x0024BA	0x0024BB	0x0024BC	0x0024BD	0x0024BE	0x0024BF	0x0024C0	0x0024C1	0x0024C2	0x0024C3	0x0024C4	0x0024C5	0x0024C6	0x0024C7	0x0024C8	0x0024C9	0x0024CA	0x0024CB	0x0024CC	0x0024CD	0x0024CE	0x0024CF	0x0024D0	0x0024D1	0x0024D2	0x0024D3	0x0024D4	0x0024D5	0x0024D6	0x0024D7	0x0024D8	0x0024D9	0x0024DA	0x0024DB	0x0024DC	0x0024DD	0x0024DE	0x0024DF	0x0024E0	0x0024E1	0x0024E2	0x0024E3	0x0024E4	0x0024E5	0x0024E6	0x0024E7	0x0024E8	0x0024E9	0x0024EA	0x0024EB	0x0024EC	0x0024ED	0x0024EE	0x0024EF	0x0024F0	0x0024F1	0x0024F2	0x0024F3	0x0024F4	0x0024F5	0x0024F6	0x0024F7	0x0024F8	0x0024F9	0x0024FA	0x0024FB	0x0024FC	0x0024FD	0x0024FE	0x0024FF
-----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------