



CPE 323 Introduction to Embedded Computer Systems: MSP430: Assembly Language and C

Aleksandar Milenkovic

Electrical and Computer Engineering
The University of Alabama in Huntsville

milenka@ece.uah.edu

<http://www.ece.uah.edu/~milenka>

Assembly Language and C

- How a high-level language uses low-level language features?
- C: Used in system programming, device drivers, ...
- Use of addressing modes by compilers
- Parameter passing in assembly language
- Local storage

C and the MSP430

- Compiler and the MSP430 instruction set
- C data types and implementation
- Storage classes
- Functions and parameters
- Pointers

Compiling a C Program: Example #1

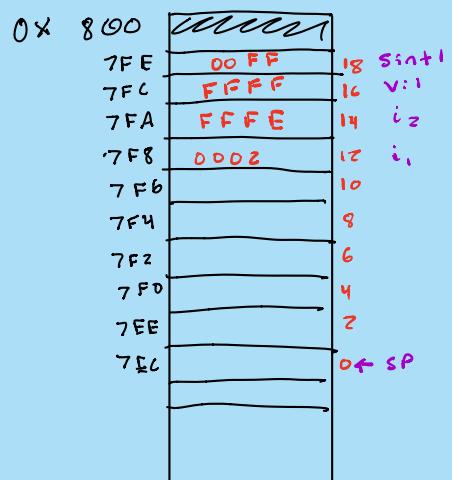
```
#include <msp430.h>

int main(void) {
    int i1, i2;
    unsigned int ui1;
    short int sint1;
    long int lint2;
    int a[4];           // int array, 4 elements
    // Stop watchdog timer to prevent time out reset
    WDTCTL = WDTPW + WDTHOLD;
    i1 = 2; i2 = -2;
    ui1=65535;
    sint1=127;
    lint2=128243;
    a[0]=20; a[1]=9;  // a[20, 9, ...]
    return 0;
}
```

Example #1 Compiler Generated List File

(TI Compiler, optimization: OFF, suppress debug symbols)

MSP430 Assembler PC v20.2.2 Mon Sep 21 09:17:08 2020
Copyright (c) 2003-2018 Texas Instruments Incorporated



Example #1 Compiler Generated List File (TI Compiler, no optimization)

No Assembly Errors, No Assembly Warnings



Example #2: demoC2ASM.c

```
#include <msp430.h>

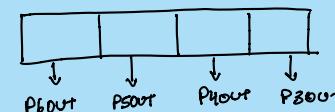
int main(void) {
    WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

    unsigned int i = 0; // switches to unsigned long int
    unsigned char ch;
    unsigned long int sum = 0;

    for(i=0; i<10; i++) sum += i;
    P3OUT = (unsigned char) sum;
    P4OUT = (unsigned char) (sum >> 8);

    ch=P1IN;
    switch(ch) {
        case 0: P2OUT=0x01; break;
        case 1: P2OUT=0x02; break;
        default: P2OUT=0x80;
    }

    return 0;
}
```



Example #2: List File (1)

MSP430 Assembler PC v20.2.2 Mon Sep 21 09:19:40 2020

```
1          ;*****
2          ;* MSP430 G3 C/C++ Codegen
*          ;* Date/Time created: Mon Sep 21 09:19:40 2020
*          ;*****
5          .compiler_opts --abi=eabi --diag_wrap=off --hll_source=on --
mem_model:code=small --mem_model:d
6          ;          C:\ti\ccs1010\ccs\tools\compiler\ti-cgt-msp430_20.2.2.LTS\bin\opt430.exe
C:\\Users\\milenka\\A
7 000000          .sect    ".text:main"
8          .clink
9          .global main
10         ;-----
11         ; 7 | int main(void) {
12         ;-----
13
14         ;*****
15         ;* FUNCTION NAME: main
16         ;*
17         ;*   Regs Modified      : SP,SR,r12,r13,r14,r15
18         ;*   Regs Used         : SP,SR,r12,r13,r14,r15
19         ;*   Local Frame Size  : 0 Args + 0 Auto + 0 Save = 0 byte
20         ;*****
21 000000          main:
22         ;*
23         ;-----
24         ; 8 | WDTCTL = WDTPW | WDTHOLD;          // stop watchdog timer
25         ; 10 | unsigned int i = 0;
26         ; 11 | unsigned char ch;
27         ;-----
28 000000 40B2          MOV.W      #23168,&WDTCTL+0      ; [] |8|
000002 5A80
000004 0000!
```

Example #2: List File (2)

```

29 ; 12 | unsigned long int sum = 0;          R15   R13
30 ; 12 | 
31 ; 
32 000006 430F      MOV.W    #0,r15          ; [] |12|
33 000008 430D      MOV.W    #0,r13          ; [] |12|
34 ; 
35 ; 14 | for(i=0; i<10; i++) sum += i;
36 ; 
37 00000a 430E      MOV.W    #0,r14          ; [] |14|
38 00000c 903E      CMP.W    #10,r14         ; [] |14|
00000e 000A
39 000010 2C06      JHS      $C$L2          ; [] |14|
40 ; [] |14|
41 ;* -----
42 ;* BEGIN LOOP $C$L1
43 ;*
44 ;* Loop source line      : 14
45 ;* Loop closing brace source line : 14
46 ;* Known Minimum Trip Count : 1
47 ;* Known Maximum Trip Count : 4294967295
48 ;* Known Max Trip Count Factor : 1
49 ;* -----
50 000012 5EOF      $C$L1:
51 000012 5EOF      ADD.W    r14,r15          ; [] |14|
52 000014 630D      ADDC.W   #0,r13          ; [] |14|
53 000016 531E      ADD.W    #1,r14          ; [] |14|
54 000018 903E      CMP.W    #10,r14         ; [] |14|
00001a 000A
55 00001c 2BFA      JLO      $C$L1          ; [] |14|
56 ; [] |14|
57 ;* -----
58 00001e           $C$L2:
59 ; 
60 ; 15 | P3OUT = (unsigned char) sum;
61 ; 
62 00001e 4FC2      MOV.B    r15,&PBOUT_L+0    ; [] |15|
000020 0000!

```

Example #2: List File (3)

```
63      ;-----
64      ; 16 | P4OUT = (unsigned char) (sum >> 8);
65      ;-----
66 000022 108F      SWPB      r15          ; [] |16|
67 000024 4FC2      MOV.B     r15,&PBOUT_H+0 ; [] |16|
000026 0000!
68      ;-----
69      ; 18 | ch=P1IN;
70      ;-----
71 000028 425F      MOV.B     &PAIN_L+0,r15 ; [] |18|
00002a 0000!
72      ;-----
73      ; 19 | switch(ch) {
74      ; 20 |   case 0: P2OUT=0x01; break;
75      ; 21 |   case 1: P2OUT=0x02; break;
76      ;-----
77 00002c 4F4F      MOV.B     r15,r15      ; [] |19|
78 00002e 930F      TST.W    r15          ; [] |19|
79 000030 2409      JEQ      $C$L4        ; [] |19|
80          ; [] |19|
81      ;* ----- *
82 000032 831F      SUB.W    #1,r15      ; [] |19|
83 000034 2404      JEQ      $C$L3        ; [] |19|
84          ; [] |19|
85      ;* ----- *
86      ;-----
87      ; 22 | default: P2OUT=0x80;
88      ;-----
89 000036 40F2      MOV.B     #128,&PAOUT_H+0 ; [] |22|
000038 0080
00003a 0000!
90 00003c 3C05      JMP      $C$L5        ; [] |23|
91          ; [] |23|
```

Example #2: List (4)

```
92          ;* -----
93 00003e      $C$L3:
94 00003e 43E2      MOV.B      #2,&PAOUT_H+0      ; [] |21|
000040 0000!
95 000042 3C02      JMP      $C$L5      ; [] |21|
96          ; [] |21|
97          ;* -----
98 000044      $C$L4:
99 000044 43D2      MOV.B      #1,&PAOUT_H+0      ; [] |20|
000046 0000!
100         ;* -----
101 000048      $C$L5:
102         ;
103         ; 25 | return 0;
104         ;
105 000048 430C      MOV.W      #0,r12      ; [] |25|
106 00004a 4130      RET      ; []
107          ; []
108          ;*****
109          ;* UNDEFINED EXTERNAL REFERENCES
110          ;*****
111          .global PAIN_L
112          .global PAOUT_H
113          .global PBOUT_L
114          .global PBOUT_H
115          .global WDTCTL
116
117          ;*****
118          ;* BUILD ATTRIBUTES
119          ;*****
120          .battr "TI", Tag_File, 1, Tag_LPM_INFO(1)
121          .battr "TI", Tag_File, 1,
```

Example #2: List File (5)

```
Tag_PORTS_INIT_INFO("012345678901ABCDEFGHIJ0000000000001111000000000
122 .battr "TI", Tag_File, 1, Tag_LEA_INFO(1)
123 .battr "TI", Tag_File, 1, Tag_HW_MPY32_INFO(2)
124 .battr "TI", Tag_File, 1, Tag_HW_MPY_ISR_INFO(1)
125 .battr "TI", Tag_File, 1, Tag_HW_MPY_INLINE_INFO(1)
126 .battr "mspabi", Tag_File, 1, Tag_enum_size(3)

No Assembly Errors, No Assembly Warnings
MSP430 Assembler PC v20.2.2 Mon Sep 21 09:19:40 2020
LABEL VALUE DEFN REF
$C$L1 000012+ 51 55
$C$L2 00001e+ 62 39
$C$L3 00003e+ 94 83
$C$L4 000044+ 99 79
$C$L5 000048+ 105 90 95
.MSP430 000001 0
.MSP4619 000000 0
.msp430 000001 0
.msp4619 000000 0
PAIN_L REF 71 111
PAOUT_H REF 89 94 99 112
PBOUT_H REF 67 114
PBOUT_L REF 62 113
WDTCTL REF 28 115
__TI_ASSEMBLER_VERSION__ 13134d2 0
__TI_EABI__ 000001 0
main 000000+ 28 9
```

C Data Types

| Data type | Size | Range | Alignment |
|--------------------|---------|--------------------------|------------------------|
| bool | 8 bits | 0 to 1 | ① any address in space |
| char | 8 bits | to 255 | 1 |
| signed char | 8 bits | -128 to 127 | 1 |
| unsigned char | 8 bits | 0 to 255 | 1 |
| signed short | 16 bits | -32768 to 32767 | 2 |
| unsigned short | 16 bits | 0 to 65535 | 2 |
| signed int | 16 bits | -32768 to 32767 | 2 |
| unsigned int | 16 bits | 0 to 65535 | 2 |
| signed long | 32 bits | - 2^{31} to $2^{31}-1$ | 2 |
| unsigned long | 32 bits | 0 to $2^{32}-1$ | 2 |
| signed long long | 64 bits | - 2^{63} to $2^{63}-1$ | 2 |
| unsigned long long | 64 bits | 0 to $2^{64}-1$ | 2 |
| float | 32 bits | | 2 |
| double | 32 bits | | 2 (*) |
| double | 64 bits | | |

C Data Types, cont'd

- Local variables
 - Defined inside a function
 - Cannot be accessed from outside the function
 - Normally lost when a return from the function is made
- Global variables
 - Defined outside a function
 - Can be accessed both from inside and outside the function
- Variables defined in a block exist only within that block

```
int i; /*global variable, visible to everything from this point*/
void function_1(void) /*A function with no parameters*/
{
    int k; /*Integer k is local to function_1*/
    {
        int q; /*Integer q exists only in this block*/
        int j; /*Integer j is local and not the same as j in main*/
    }
}
void main(void)
{
    int j; /*Integer j is local to this block within function main*/
} /*This is the point at which integer j ceases to exist*/
```

Storage Class Specifiers

- **auto** → Default
 - Variable is no longer required once a block has been left; Default
- **register**
 - Ask compiler to allocate the variable to a register
 - Also is automatic
 - Cannot be accessed by means of pointers
- **static** will stay alive when you leave function
 - Allows local variable to retain its value when a block is reentered
 - Initialized only once, by the compiler!
- **extern**
 - Indicates that the variable is defined outside the block
 - The same global variable can be defined in more than one module

Storage Class Modifiers

- **volatile**
 - To define variables that can be changed externally
 - Compiler will not put them in registers
 - Think about Status Registers ! *peripherals*
- **const**
 - Variable may not be changed during the execution of a program
 - Cannot be changed unintentionally,
but CAN be changed externally
(as a result of an I/O, or OS operations external to the C program)
- Type conversion
 - In C, done either automatically or explicitly (casting)

Compiling a C Program: Example #3

```
int main(void) {  
    volatile int i1, i2;  
    volatile unsigned int ui1;  
    volatile short int sint1;  
    volatile long int lint2;  
    volatile int a[4];  
    // Stop watchdog timer to prevent time out reset  
    WDTCTL = WDTPW + WDTHOLD;  
    i1 = 2; i2 = -2;  
    ui1=65535;  
    sint1=127;  
    lint2=128243;  
    a[0]=20; a[1]=9;  
    return 0;  
}
```

Example #3 Compiler Generated List File (no optimization)

DC8
↑
Constants

```
C:\Documents and Settings\Aleksandar\My Documents\Work\teaching\cpe323-08F\tutorial\test_dtypes.c
1           #include "io430.h"
           \
           \           In  segment DATA16_AN, at 0x120
           \           union <unnamed> volatile __data16 __A_WDTCTL
           \           \           _A_WDTCTL:
           \           000000           DS8 2 Allocate storage

           \
           \           In  segment CODE, align 2
2           int main( void ) {
           \
           \           main:
           \           000000  31801400    SUB.W  #0x14, SP
           3           volatile int i1, i2;
           4           volatile unsigned int uil;
           5           volatile short int sint1;
           6           volatile long int lint2;
           7           volatile int a[4];
           8           // Stop watchdog timer to prevent time out reset
           9           WDTCTL = WDTPW + WDTHOLD;
           \
           \           000004  B240805A2001 MOV.W  #0x5a80, &0x120
           10          i1 = 2; i2 = -2;
           \
           \           00000A  A1430000  MOV.W  #0x2, 0(SP)
           \
           \           00000E  B140FEFF0200 MOV.W  #0xffffe, 0x2(SP)
           11          uil=65535;
           \
           \           000014  B1430400  MOV.W  #0xffff, 0x4(SP)
           12          sint1=127;
           \
           \           000018  B1407F000600 MOV.W  #0x7f, 0x6(SP)
           13          lint2=128243;
           \
           \           00001E  B140F3F40800 MOV.W  #0xf4f3, 0x8(SP)
           \
           \           000024  91430A00  MOV.W  #0x1, 0xa(SP)
```

Example #3 Compiler Generated List File (no optimization)

```
14          a[0]=20; a[1]=9;
\ 000028    B14014000C00 MOV.W   #0x14, 0xc(SP)
\ 00002E    B14009000E00 MOV.W   #0x9, 0xe(SP)
15          return 0;
\ 000034    0C43          MOV.W   #0x0, R12
\ 000036    31501400      ADD.W   #0x14, SP
\ 00003A    3041          RET
\ 00003C          REQUIRE _A_WDTCTL
16      }
Maximum stack usage in bytes:
```

```
Function CSTACK
-----
main      22
```

Segment part sizes:

```
Function/Label Bytes
-----
_A_WDTCTL    2
main         60
```

60 bytes in segment CODE
2 bytes in segment DATA16_AN

60 bytes of CODE memory
0 bytes of DATA memory (+ 2 bytes shared)

Errors: none

Warnings: none

Example #4: Factorial

```
#include "stdio.h"
#include "io430.h"

int fact(int n);

int main(void) {

    int n = 5;

    int nf;

    nf = fact(n);

    printf("n=%d, nf=%d\n", n, nf);

    return 0;
}

int fact(int n) {

    if(n>1) return n*fact(n-1);
    else return 1;
}
```

Example #4: Factorial, List File

```
1      # include "stdio.h"
2      #include "io430.h"
4      int fact(int n);
\\                                         In  segment CODE, align 2
6      int main(void) {
7          main:
8          000000 0A12      PUSH.W  R10
9          000002 0B12      PUSH.W  R11
10         000004 3A400500  MOV.W   #0x5, R10
11
12         int nf;
13
14         nf = fact(n);
15
16         printf("n=%d, nf=%d\n", n, nf);
17         000010 0B12      PUSH.W  R11
18         000012 0A12      PUSH.W  R10
19         000014 3C40....  MOV.W   #`?<Constant "n=%d, nf=%d\n">` , R12
20         000018 B012....  CALL    #printf
21
22         return 0;
23         00001C 0C43      MOV.W   #0x0, R12
24         00001E 2152      ADD.W   #0x4, SP
25         000020 3B41      POP.W   R11
26         000022 3A41      POP.W   R10
27         000024 3041      RET
```

Example #4: Factorial, List File (cont'd)

```
19         int fact(int n) {
          \
          \ 000000 0A12      PUSH.W  R10
          \ 000002 0A4C      MOV.W   R12, R10
20
21         if(n>1)  return n*fact(n-1);
          \
          \ 000004 2A93      CMP.W   #0x2, R10
          \ 000006 0E38      JL      ??fact_0
          \
          \ 000008 0C4A      MOV.W   R10, R12
          \ 00000A 3C53      ADD.W   #0xffff, R12 // subtracting 1
          \
          \ 00000C B012....  CALL    #fact
          \
          \ 000010 0212      PUSH.W  SR
          \
          \ 000012 32C2      DINT
          \
          \ 000014 824A3001  MOV.W   R10, &0x130
          \
          \ 000018 824C3801  MOV.W   R12, &0x138
          \
          \ 00001C 1C423A01  MOV.W   &0x13a, R12
          \
          \ 000020 3241      POP.W   SR
          \
          \ 000022 013C      JMP     ??fact_1
22         else return 1;
          \
          \ 000024 1C43      ??fact_0:
          \
          \ 000026 3A41      ??fact_1:
          \
          \ 000028 3041      POP.W   R10
23         }
          \
          \ sorted
          \
          \ ?<Constant "n=%d, nf=%d\n">:
          \
          \ 000000 6E3D25642C20 DC8 "n=%d, nf=%d\012"
          \
          \ 6E663D25640A
          \
          \ 00
```

Functions and Parameters

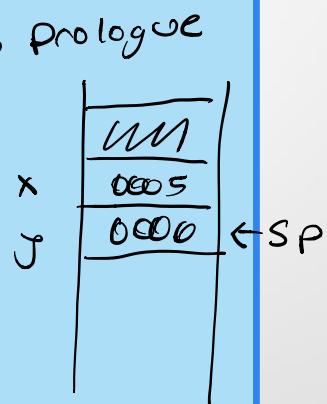
```
#include "io430.h"
void swapbyv(int a, int b);
void swapbyr(int *a, int *b);
int main( void )
{
    // Stop watchdog timer to prevent time out reset
    WDTCTL = WDTPW + WDTHOLD;
    int x = 5;
    int y = 6;
    // pass parameters by value
    swapbyv(x, y);
    // pass parameters by reference
    swapbyr(&x, &y);

    return 0;
}
```

```
void swapbyv(int a, int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}

void swapbyr(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

Functions and Parameters



Functions and Parameters

```

\                                     In  segment CODE,
align 2
 32          void swapbyv(int a, int b) {
\             swapbyv:
 33             int temp;
 34
 35             temp = a;
\ 000000  0F4C          MOV.W   R12, R15
 36             a = b;
\ 000002  0C4D          MOV.W   R13, R12
 37             b = temp;
\ 000004  0D4F          MOV.W   R15, R13
 38           }
\ 000006  3041          RET
 39

\                                     In  segment CODE,
align 2
 40          void swapbyr(int *a, int *b) {
\             swapbyr:
 41             int temp;
 42
 43             temp = *a;
\ 000000  2F4C          MOV.W   @R12, R15
 44             *a = *b;
\ 000002  AC4D0000        MOV.W   @R13, 0(R12)
 45             *b = temp;
\ 000006  8D4F0000        MOV.W   R15, 0(R13)
 46           }
\ 00000A  3041          RET

```

Maximum stack usage in bytes:

| Function | CSTACK |
|------------|--------|
| main | 6 |
| -> swapbyv | 6 |
| -> swapbyr | 6 |
| swapbyr | 2 |
| swapbyv | 2 |

Segment part sizes:

| Function/Label | Bytes |
|----------------|-------|
| _A_WDTCTL | 2 |
| main | 46 |
| swapbyv | 8 |
| swapbyr | 12 |

66 bytes in segment CODE

2 bytes in segment DATA16_AN

66 bytes of CODE memory

0 bytes of DATA memory (+ 2 bytes shared)

Pointers and C

```
#include "io430.h"
#include "stdio.h"
int main( void )
{
    int x = 5; // an integer x
    int *p_x; // a pointer to int
    int y1; // an integer y1 (uninitialized)
    long int y2, y3; // long integers y2, y3
    long int *p_y2; // a pointer to long integer
    char mya[20] = "hello world, cpe323!"; // character array
    char *p_mya; // pointer to character
    WDTCTL = WDTPW + WDTHOLD; // stop WDT
    p_x = &x; // p_x points to x
    y1 = 10 + x; // new value to y1
    y2 = -1;
    p_y2 = &y2; // pointer p_y2 points to y2
    y3 = 10 + *p_y2;
    p_mya = mya; // p_mya points to array mya
    p_mya = p_mya + 3;
    // display addresses and variables in terminal i/o
    printf("a.x=%x, x=%x\n", &x, x);
    printf("a.p_x=%x, p_x=%x\n", &p_x, p_x);
    printf("a.y1=%x, y1=%x\n", &y1, y1);
    printf("a.y2=%x, y2=%lx\n", &y2, y2);
    printf("a.y3=%x, y3=%lx\n", &y3, y3);
    printf("a.p_y2=%x, p_y2=%x\n", &p_y2, p_y2);
    printf("a.mya=%x, mya=%s\n", &mya, mya);
    printf("a.p_mya=%x, p_mya=%x\n", &p_mya, p_mya);
    return 0;
}
```

Pointers and C, cont'd

```
1      #include "io430.h"                                In segment DATA16_AN, at 0x120
\      union <unnamed> volatile __data16 __A_WDTCTL
\          __A_WDTCTL:
\ 000000      DS8 2
2      #include "stdio.h"
3
\          In segment CODE, align 2
4      int main(void) {
\          main:
\ 000000  31802600      SUB.W  #0x26, SP
5          // Stop watchdog timer to prevent time out reset
6          WDTCTL = WDTPW + WDTHOLD;
\ 000004  B240805A2001 MOV.W  #0x5a80, &0x120
7          int x = 5; // an integer x
\ 00000A  B140050000000 MOV.W  #0x5, 0(SP)
8          int *p_x; // a pointer to int
9          int y1; // an integer y1 (uninitialized)
10         long int y2, y3; // long integers y2, y3
11         long int *p_y2; // a pointer to long integer
12         char mya[20] = "hello world, cpe323!"; // character array
\ 000010  0C41          MOV.W  SP, R12
\ 000012  3C501200      ADD.W  #0x12, R12
\ 000016  3E40....      MOV.W  #`?<Constant "hello world, cpe323!">`, R14
\ 00001A  3D401400      MOV.W  #0x14, R13
\ 00001E  B012....      CALL   #?CopyMemoryBytes
13         char *p_mya; // pointer to character
14
15         p_x = &x; // p_x points to x
\ 000022  0F41          MOV.W  SP, R15
\ 000024  814F0800      MOV.W  R15, 0x8(SP)
```

Pointers and C, cont'd

```
16          y1 = 10 + x;      // new value to y1
\ 000028  2F41          MOV.W  @SP, R15
\ 00002A  3F500A00      ADD.W  #0xa, R15
\ 00002E  814F0600      MOV.W  R15, 0x6(SP)

17          y2 = -1;
\ 000032  B1430A00      MOV.W  #0xffff, 0xa(SP)
\ 000036  B1430C00      MOV.W  #0xffff, 0xc(SP)
18          p_y2 = &y2;      // pointer p_y2 points to y2
\ 00003A  0F41          MOV.W  SP, R15
\ 00003C  3F500A00      ADD.W  #0xa, R15
\ 000040  814F0400      MOV.W  R15, 0x4(SP)

19          y3 = 10 + *p_y2;
\ 000044  1F410400      MOV.W  0x4(SP), R15
\ 000048  2E4F          MOV.W  @R15, R14
\ 00004A  1F4F0200      MOV.W  0x2(R15), R15
\ 00004E  3E500A00      ADD.W  #0xa, R14
\ 000052  0F63          ADDC.W #0x0, R15
\ 000054  814E0E00      MOV.W  R14, 0xe(SP)
\ 000058  814F1000      MOV.W  R15, 0x10(SP)

20          p_mya = mya;    // p_mya points to array mya
\ 00005C  0F41          MOV.W  SP, R15
\ 00005E  3F501200      ADD.W  #0x12, R15
\ 000062  814F0200      MOV.W  R15, 0x2(SP)

21          p_mya = p_mya + 3;
\ 000066  B15003000200 ADD.W  #0x3, 0x2(SP)
```

Example #5: Pointers and Pointer Arithmetic

Consider the following C program.

Assume that the register SP at the beginning points to 0xA00.

Assume all variables are allocated on the stack, and in the order as they appear in the program (a, b, c, mych, pli, pi).
 ASCII code for character '0' is 48 (0x30).

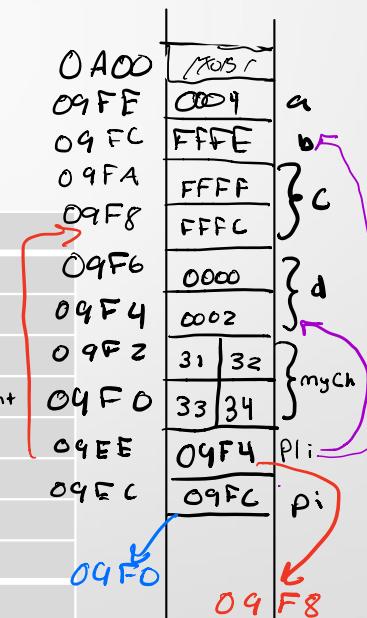
First element @ lowest address

```

1 int main( void ) {
2   volatile int a = 4, b = -2;
3   volatile long int c = -4, d = 2; FFFF FFFF
4   volatile char mych = {'4', '3', '2', '1'};
5   volatile long int *pli = &d; // Var is pointer, to type long int
6   volatile int *pi = &b;
7   pli = pli + 1; Pointer Arithmetic 04F8 1x4
8   pi = pi - 6;      04FC - 6x2 09F0 (34)
9   *pi = a + *pi; pi
10 }
Line 7: pli++; // next object in sequence of that type
        ↘ 1x4
  
```

Line 8: $pi = pi - 6 \rightarrow 6 \times 2 = 12$

Line 9: $*pi = a + *pi; \quad \begin{array}{r} 3334 \\ + 0004 \\ \hline 3338 \end{array}$



Example #5 (cont'd)

| # | Question? | Value/Address |
|----|--|---------------|
| 1 | The number of bytes allocated on the stack for the variables declared in line 2. <i>Volatile int a=4, b=-2</i> | 4 bytes |
| 2 | The number of bytes allocated on the stack for the character array declared in line 4. <i>Volatile char mych = {'4','3','2','1'}</i> | 4 bytes |
| 3 | The number of bytes allocated on the stack for all variables declared in lines 2-6. | 20 bytes |
| 4 | Value of mych[0] after initialization performed in line 4. | '4' or 34 |
| 5 | Address of variable b (&b). | 09FC |
| 7 | Value of pli at the moment after the statement in line 5 is executed. | 09F4 |
| 8 | Value of pli at the moment after the statement in line 7 is executed. | 09F8 |
| 9 | Value of pi at the moment after the statement in line 8 is executed. | 09FD |
| 10 | Value of mych[0] at the moment after the statement in line 9 is executed. | 0x38 '8' |

- Volatile int a[4] = { 0, 1, 2, 3 }

```
Volatile int *pa;
```

```
pa = a; // address of A
```

```
pa = pa + 2; // 0x07F8 + 2*2 = 0x07FC
```

| | |
|-----|--------|
| | 111111 |
| 800 | 0003 |
| 7FE | 0002 |
| 7FC | 0001 |
| 7FA | 0000 |
| 7F8 | 07F8 |
| | 07FC |
| | pa |

```
char mycn[5] = { 0, 1, 2, 'A', '3' };
```

→ skipped / unused

| | |
|----|----|
| | 42 |
| 41 | 02 |
| 01 | 00 |