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

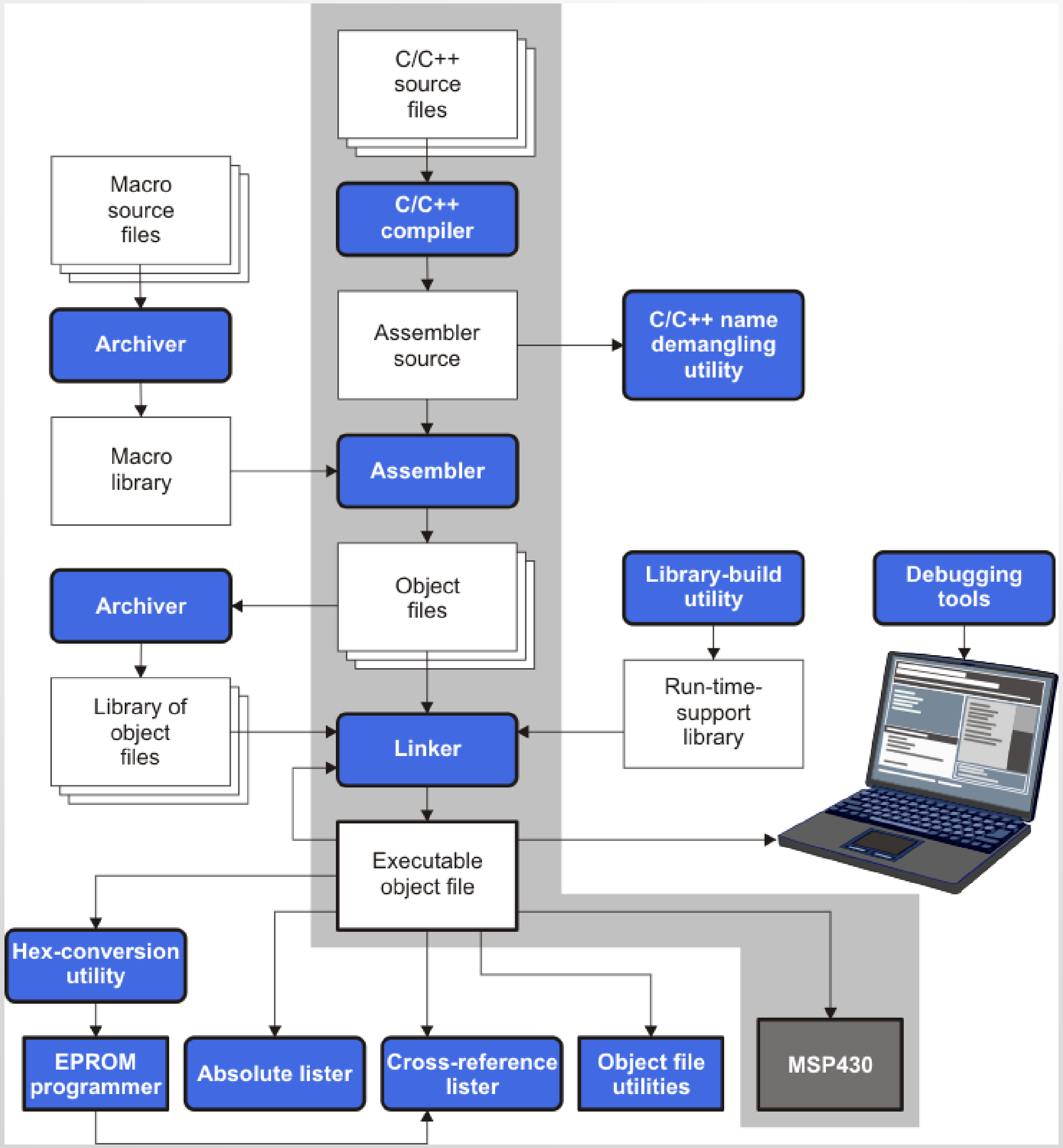
- How a high-level language uses low-level language features?
- C: Used in system programming, device drivers, …
- Use of addressing modes by compilers
- Parameter passing in assembly language
- Local storage

C and the MSP430

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

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Software Design Flow



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];
	// 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 #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
CDataAllocationDemo.asm                                              PAGE    1

       1              ;******************************************************************************
       2              ;* MSP430 G3 C/C++ Codegen                                         PC v20.2.2.LTS *
       3              ;* Date/Time created: Mon Sep 21 09:17:08 2020                                  *
       4              ;******************************************************************************
       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\acpia430.exe -@C:\\Users\\milenk
       7 000000                .sect    ".text:main"
       8                      .clink
       9                      .global main
      10              ;----------------------------------------------------------------------
      11              ;  25 | int main(void) {
      12              ;  26 | int i1, i2;
      13              ;  27 | unsigned int ui1;
      14              ;  28 | short int sint1;
      15              ;  29 | long int lint2;
      16              ;  30 |  int a[4];
      17              ;  31 | // Stop watchdog timer to prevent time out reset
      18              ;----------------------------------------------------------------------
      19
      20              ;*****************************************************************************
      21              ;* FUNCTION NAME: main                                                      *
      22              ;*                                                                          *
      23              ;*   Regs Modified     : SP,SR,r12                                          *
      24              ;*   Regs Used         : SP,SR,r12                                          *
      25              ;*   Local Frame Size  : 0 Args + 20 Auto + 0 Save = 20 byte                *
      26              ;*****************************************************************************
      27 000000        main:
      28              ;* --------------------------------------------------------------------------*
      29 000000 8031          SUB.W     #20,SP                ; []
         000002 0014
      30              ;----------------------------------------------------------------------
      31              ;  32 | WDTCTL = WDTPW + WDTHOLD;
      32              ;----------------------------------------------------------------------
      33 000004 40B2          MOV.W     #23168,&WDTCTL+0      ; [] |32|
         000006 5A80
         000008 0000!
      34              ;----------------------------------------------------------------------
      35              ;  33 | i1 = 2; i2 = -2;
      36              ;----------------------------------------------------------------------
      37 00000a 43A1          MOV.W     #2,12(SP)             ; [] |33|
         00000c 000C
      38 00000e 40B1          MOV.W     #65534,14(SP)         ; [] |33|
         000010 FFFE
         000012 000E
      39              ;----------------------------------------------------------------------
      40              ;  34 | ui1=65535;
      41              ;----------------------------------------------------------------------
      42 000014 43B1          MOV.W     #65535,16(SP)         ; [] |34|
         000016 0010
      43              ;----------------------------------------------------------------------
      44              ;  35 | sint1=127;
      45              ;----------------------------------------------------------------------
      46 000018 40B1          MOV.W     #127,18(SP)           ; [] |35|
         00001a 007F
         00001c 0012
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)

```
CDataAllocationDemo.asm                                             PAGE    2

      47              ;----------------------------------------------------------------------
      48              ;  36 | lint2=128243;
      49              ;----------------------------------------------------------------------
      50 00001e 40B1          MOV.W     #62707,8(SP)          ; [] |36|
         000020 F4F3
         000022 0008
      51 000024 4391          MOV.W     #1,10(SP)             ; [] |36|
         000026 000A
      52              ;----------------------------------------------------------------------
      53              ;  37 | a[0]=20; a[1]=9;
      54              ;----------------------------------------------------------------------
      55 000028 40B1          MOV.W     #20,0(SP)             ; [] |37|
         00002a 0014
         00002c 0000
      56 00002e 40B1          MOV.W     #9,2(SP)              ; [] |37|
         000030 0009
         000032 0002
      57              ;----------------------------------------------------------------------
      58              ;  38 | return 0;
      59              ;----------------------------------------------------------------------
      60 000034 430C          MOV.W     #0,r12                ; [] |38|
      61 000036 5031          ADD.W     #20,SP                ; []
         000038 0014
      62 00003a 4130          RET       ; []
      63                      ; []
      64              ;***************************************************************
      65              ;* UNDEFINED EXTERNAL REFERENCES                               *
      66              ;***************************************************************
      67                      .global WDTCTL
      68
      69              ;****************************************************************
      70              ;* BUILD ATTRIBUTES                                             *
      71              ;****************************************************************
      72                      .battr "TI", Tag_File, 1, Tag_LPM_INFO(1)
      73                      .battr "TI", Tag_File, 1,
Tag_PORTS_INIT_INFO("012345678901ABCDEFGHIJ00000000000011110000000
      74                      .battr "TI", Tag_File, 1, Tag_LEA_INFO(1)
      75                      .battr "TI", Tag_File, 1, Tag_HW_MPY32_INFO(2)
      76                      .battr "TI", Tag_File, 1, Tag_HW_MPY_ISR_INFO(1)
      77                      .battr "TI", Tag_File, 1, Tag_HW_MPY_INLINE_INFO(1)
      78                      .battr "mspabi", Tag_File, 1, Tag_enum_size(3)

No Assembly Errors, No Assembly Warnings
```

Variables allocated on the program stack for CDataAllocationDemo.c when executed on MSP430F5529

Address	Memory[15:0] [hex]	Offset relative to current SP	Variable
0x4400	---		*Original Top of the Stack*
0x43FE	0x00FF	18	sint1
0x43FC	0xFFFF	16	ui1
0x43FA	0xFFFE	14	i2
0x43F8	0x0002	12	i1
0x43F6	0x0001	10	lint2 (upper)
0x43F4	0xF4F3	8	lint2 (lower)
0x43F2	-	6	a[3]
0x43F0	-	4	a[2]
0x43EE	0x0009	2	a[1]
0x43EC	0x0014	0 <= SP	a[0]

Example #2: demoC2ASM.c

```
#include <msp430.h>

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

    unsigned int i = 0;
    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;
}
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

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                                    PC v20.2.2.LTS
    *
      3              ;* Date/Time created: Mon Sep 21 09:19:40 2020                              *
      4              ;******************************************************************************
      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!
```

LaCASA

Example #2: List File (2)

```
           ;----------------------------------------------------------------------
           ;  12 | unsigned long int sum = 0;
           ;----------------------------------------------------------------------
000006 430F          MOV.W     #0,r15                ; [] |12|
000008 430D          MOV.W     #0,r13                ; [] |12|
           ;----------------------------------------------------------------------
           ;  14 | for(i=0; i<10; i++) sum += i;
           ;----------------------------------------------------------------------
00000a 430E          MOV.W     #0,r14                ; [] |14|
00000c 903E          CMP.W     #10,r14               ; [] |14|
00000e 000A
000010 2C06          JHS       $C$L2                 ; [] |14|
                                                     ; [] |14|
           ;* ----------------------------------------------------------------------*
           ;*   BEGIN LOOP $C$L1
           ;*
           ;*   Loop source line                : 14
           ;*   Loop closing brace source line  : 14
           ;*   Known Minimum Trip Count        : 1
           ;*   Known Maximum Trip Count        : 4294967295
           ;*   Known Max Trip Count Factor     : 1
           ;* ----------------------------------------------------------------------*
000012       $C$L1:
000012 5E0F          ADD.W     r14,r15               ; [] |14|
000014 630D          ADDC.W    #0,r13                ; [] |14|
000016 531E          ADD.W     #1,r14                ; [] |14|
000018 903E          CMP.W     #10,r14               ; [] |14|
00001a 000A
00001c 2BFA          JLO       $C$L1                 ; [] |14|
                                                     ; [] |14|
           ;* ----------------------------------------------------------------------*
00001e       $C$L2:
           ;----------------------------------------------------------------------
           ;  15 | P3OUT = (unsigned char) sum;
           ;----------------------------------------------------------------------
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)

```
           ;* --------------------------------------------------------------------------*
00003e       $C$L3:
00003e 43E2          MOV.B     #2,&PAOUT_H+0         ; [] |21|
000040 0000!
000042 3C02          JMP       $C$L5                 ; [] |21|
                                                     ; [] |21|
           ;* --------------------------------------------------------------------------*
000044       $C$L4:
000044 43D2          MOV.B     #1,&PAOUT_H+0         ; [] |20|
000046 0000!
           ;* --------------------------------------------------------------------------*
000048       $C$L5:
           ;-----------------------------------------------------------------------
           ;  25 | return 0;
           ;-----------------------------------------------------------------------
000048 430C          MOV.W     #0,r12                ; [] |25|
00004a 4130          RET       ; []
                   ; []
           ;******************************************************************************
           ;* UNDEFINED EXTERNAL REFERENCES                                              *
           ;******************************************************************************
                   .global PAIN_L
                   .global PAOUT_H
                   .global PBOUT_L
                   .global PBOUT_H
                   .global WDTCTL

           ;*******************************************************************************
           ;* BUILD ATTRIBUTES                                                            *
           ;*******************************************************************************
                   .battr "TI", Tag_File, 1, Tag_LPM_INFO(1)
                   .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
```

MSP430 C/C++ Data Types

	Size [bits]	Align-ment	Representation	Range	
Type				**Minimum**	**Maximum**
signed char	8	8	Binary	-128	127
char	8	8	ASCII	0	255
unsigned char	8	8	Binary	0	255
bool (C99)	8	8	Binary	0 (false)	1 (true)
_Bool (C99)	8	8	Binary	0 (false)	1 (true)
bool (C++)	8	8	Binary	0 (false)	1 (true)
short, signed short	16	16	2s complement	-32,768	32,767
unsigned short	16	16	Binary	0	65,535
int, signed int	16	16	2's complement	-32,768	32,767
unsigned int	16	16	Binary	0	65,535
long, signed long	32	16	2's complement	-2,147,483,648	2,147,483,647
unsigned long	32	16	Binary	0	4,294,967,295
long long, signed long long	64	16	2's complement	-9,223,372,036, 854,775,808	9,223,372,036, 854,775,807
unsigned long long	64	16	Binary	0	18,446,744,073, 709,551,615
enum	varies	16	2's complement	varies	varies
float	32	16	IEEE 32-bit	1.175494e-38	3.40282346e+38
double	64	16	IEEE 64-bit	2.22507385e-308	1.79769313e+308

Data Sizes for MSP430 Pointers

Code and Data Model	Type	Size	Storage	Alignment
small code model	function pointer	16	16	16
large code model	function pointer	20	32	16
small data model	data pointer	16	16	16
small data model	size_t	16	16	16
small data model	ptrdiff_t	16	16	16
large data model	data pointer	20	32	16
large data model	size_t	32	32	16
large data model	ptrdiff_t	32	32	16

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

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*/
```

LaCASA

Storage Class Specifiers

- `auto`
  - 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`
  - 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 !
- `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)

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Compiling a C Program: Example #3

```
#include <msp430.h>
#include <stdio.h>

int gi = 5; // global variable, initialized to 5

char lc2uc(char *pc); // function prototype
int plus10(int i);   // function prototype

void main(void) {
  int li1 = 2;    // local var, li1=2
  char ch1 = 'a'; // local var, ch1='a'
  char ch2;       // local var, ch2 not initialized

  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  ch2 = lc2uc(&ch1);  // call lc2uc function
  li1 = li1 + gi;     // update li1
  li1 = plus10(li1);  // call plus10 function
  printf("li1=%d, gi=%d\n", li1, gi);
  printf("ch1=%c, ch2=%c\n", ch1, ch2);
}
```

```
char lc2uc(char *pc) {
    char tc;
    tc = *pc;
    if ((tc >= 'a') && (tc <= 'z')) tc = tc + ('A' - 'a'); // convert
lowercase to uppercase
    *pc = tc;                                  //
    return (tc+1);
}

int plus10(int i) {
    i = i + 10;
    gi = gi + 10;
    return 20;
}
```

LaCASA

Example #3 Compiler Generated List File (no optimization)

```
;*****************************************************************************
;* FUNCTION NAME: plus10                                                     *
;*                                                                           *
;*   Regs Modified     : SP,SR,r12                                           *
;*   Regs Used         : SP,SR,r12                                           *
;*   Local Frame Size  : 0 Args + 0 Auto + 0 Save = 0 byte                   *
;*****************************************************************************
plus10:
;* --------------------------------------------------------------------------*
;** 43    -----------------------    gi += 10;
        ADD.W     #10,&gi+0             ; [] |43|
;** 44    -----------------------    return 20;
        MOV.W     #20,r12               ; [] |44|
        RET       ; []
        ; []
     .sect ".text:lc2uc"
     .clink
     .global    lc2uc
```

Example #3 Compiler Generated List File (no optimization)

```
;******************************************************************************
;* FUNCTION NAME: lc2uc                                                       *
;*                                                                            *
;*   Regs Modified     : SP,SR,r12,r15                                        *
;*   Regs Used         : SP,SR,r12,r15                                        *
;*   Local Frame Size  : 0 Args + 0 Auto + 0 Save = 0 byte                    *
;******************************************************************************
lc2uc:
;* --------------------------------------------------------------------------*
;** 35      -----------------------    if ( (tc = *pc) < 97 || tc > 122 ) goto g3;
        MOV.B     @r12,r15                ; [] |35|
        CMP.B     #97,r15                 ; [] |35|
        JLO       $C$L1                   ; [] |35|
                                            ; [] |35|
;* --------------------------------------------------------------------------*
        CMP.B     #123,r15                ; [] |35|
        JHS       $C$L1                   ; [] |35|
                                            ; [] |35|
;* --------------------------------------------------------------------------*
;** 36      -----------------------    tc -= 32;
        SUB.B     #32,r15                 ; [] |36|
;* --------------------------------------------------------------------------*
$C$L1:
;**   -----------------------g3:
;** 37      -----------------------    *pc = tc;
        MOV.B     r15,0(r12)              ; [] |37|
;** 38      -----------------------    return (unsigned char)(tc+1);
        MOV.W     #1,r12                  ; [] |38|
        ADD.B     r15,r12                 ; [] |38|
        RET       ; []
        ; []
      .sect ".text:main"
      .clink
      .global     main
```

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;
}
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

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) {
   \                     main:
   \   000000   0A12         PUSH.W  R10
   \   000002   0B12         PUSH.W  R11
     7
     8             int n = 5;
   \   000004   3A400500     MOV.W   #0x5, R10
     9
    10             int nf;
    11
    12             nf = fact(n);
   \   000008   0C4A         MOV.W   R10, R12
   \   00000A   B012....     CALL    #fact
   \   00000E   0B4C         MOV.W   R12, R11
    13
    14             printf("n=%d, nf=%d\n", n, nf);
   \   000010   0B12         PUSH.W  R11
   \   000012   0A12         PUSH.W  R10
   \   000014   3C40....     MOV.W   #`?<Constant "n=%d, nf=%d\\n">`, R12
   \   000018   B012....     CALL    #printf
    15
    16             return 0;
   \   00001C   0C43         MOV.W   #0x0, R12
   \   00001E   2152         ADD.W   #0x4, SP
   \   000020   3B41         POP.W   R11
   \   000022   3A41         POP.W   R10
   \   000024   3041         RET
    17           }
```

LaCASA

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

```
  19           int fact(int n) {
\                      fact:
\   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
\   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;
\                      ??fact_0:
\   000024   1C43          MOV.W   #0x1, R12
\                      ??fact_1:
\   000026   3A41          POP.W   R10
\   000028   3041          RET
  23           }
\                                  In  segment DATA16_C, align 1, align-
 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;
}
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Functions and Parameters

```
8           int main( void )
  \                      main:
     9          {
  \   000000   2182          SUB.W   #0x4, SP
    10             // Stop watchdog timer to prevent time out reset
    11             WDTCTL = WDTPW + WDTHOLD;
  \   000002   B240805A2001 MOV.W   #0x5a80, &0x120
    12
    13             int x = 5;
  \   000008   B14005000200 MOV.W   #0x5, 0x2(SP)
    14             int y = 6;
  \   00000E   B14006000000 MOV.W   #0x6, 0(SP)

    19             swapbyv(x, y);
  \   000014   2D41          MOV.W   @SP, R13
  \   000016   1C410200      MOV.W   0x2(SP), R12
  \   00001A   B012....      CALL    #swapbyv

    24             swapbyr(&x, &y);
  \   00001E   0D41          MOV.W   SP, R13
  \   000020   0C41          MOV.W   SP, R12
  \   000022   2C53          ADD.W   #0x2, R12
  \   000024   B012....      CALL    #swapbyr

    29             return 0;
  \   000028   0C43          MOV.W   #0x0, R12
  \   00002A   2152          ADD.W   #0x4, SP
  \   00002C   3041          RET
  \   00002E                 REQUIRE _A_WDTCTL
    30          }
```

LaCASA

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   B14005000000 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

- For simplicity we are going to assume that SP initially points to 0x4400. In addition, we are going to assume that the variables are allocated on the stack in the order of appearance in the program.

```
int main(void) {
   volatile unsigned int a = 4, b = 2;
   volatile long int c = -4, d = 2;
   volatile char mych]4] = {‘4’, ‘3’, ‘2’, ‘1’};
   volatile long int *pli = &d;
   volatile int *pi = &b;

   pli = pli + 1;
  pi = pi – 6;
  *pi = a + *pi;
}
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Example #5: Stack

Address	Memory[15:0] [hex]	Offset relative to current SP	Variable
0x4400	---		*Original Top of the Stack*
0x43FE	0x0004	18	a
0x43FC	0x0002	16	b
0x43FA	0xFFFF	14	c, upper word
0x43F8	0xFFFC	12	c, lower word
0x43F6	0x0000	10	d, upper word
0x43F4	0x0002	8	d, lower word
0x43F2	0x3132	6	mych[3], mych[2]
0x43F0	0x3334	4	mych[1], mych[0]
0x43EE	0x43F2	2	pli
0x43EC	0x43FC	0 <= SP	pi

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Example #5 (cont'd)

#	Question?	Value/Address
1	The number of bytes allocated on the stack for the variables declared in line 02.	
2	The number of bytes allocated on the stack for the character array declared in line 04.	
3	The number of bytes allocated on the stack for all variables declared in lines 2-6.	
4	Value of mych[0] after initialization performed in line 04.	
5	Address of variable b (&b).	
7	Value of pli at the moment after the statement in line 05 is executed.	
8	Value of pli at the moment after the statement in line 08 is executed.	
9	Value of pi at the moment after the statement in line 09 is executed.	
10	Value of mych[0] at the moment after the statement in line 10 is executed.	