

CPE 323

Intro to Embedded Computer Systems

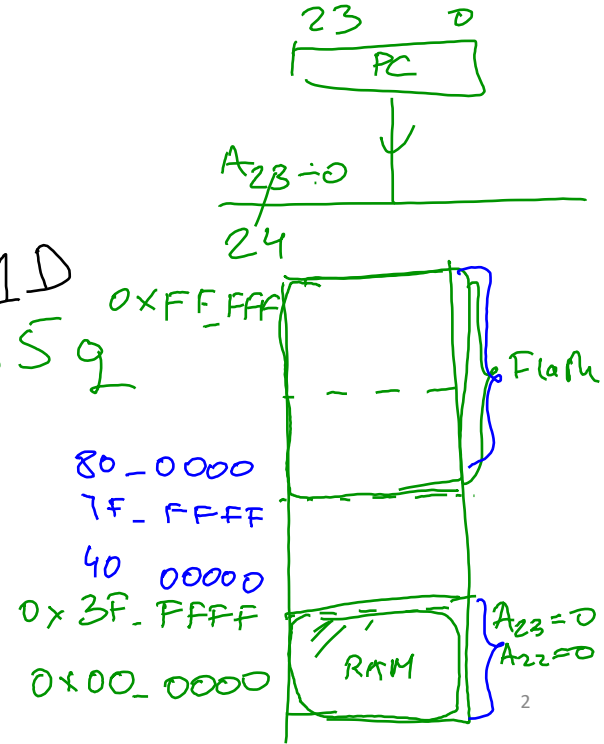
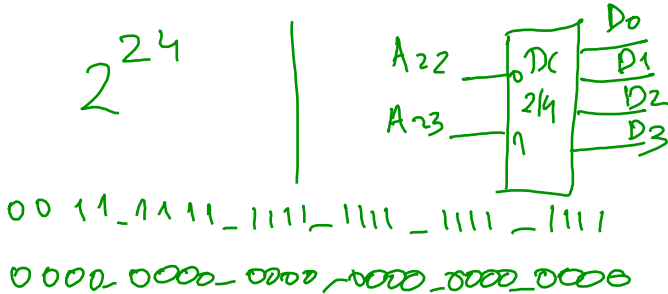
Assembly Language Programming

Aleksandar Milenkovic

milenka@uah.edu

Admin

- 8-bit
- $\boxed{0001} \boxed{1101}_6 = 0 \times 10$
- $= 035$



1a) $0x1116$ MOV 6(R5), R7 ; $R7 \leftarrow M[6+R5]$
 Move word Indexed register direct

| | | | | |
|------|-----------|---|------|---|
| | Ad B/W As | | | |
| 1116 | 4 | 5 | 0001 | 7 |
| 1118 | 0 | 0 | 0 | 6 |

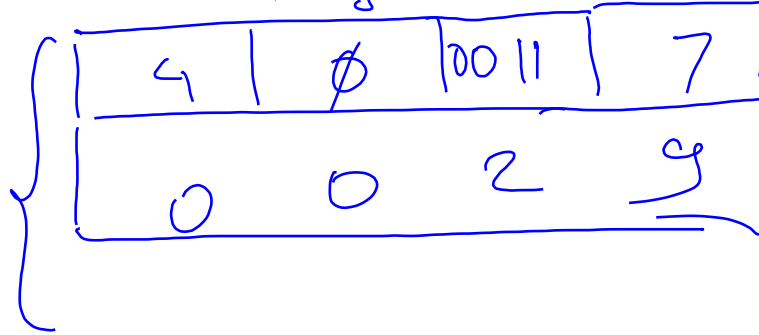
$$\begin{aligned} EA_5 &= 6 + R5 \\ &= 6 + F0D2 \\ &= 0xF0D8 \end{aligned}$$

$$R5 = 0xF0D2$$

$$\begin{aligned} R7 &\leftarrow M[0xF0D8] \\ &= 0xF014 \end{aligned}$$

MOV.B 6(R5), R7

(i) $\text{MOV } \underbrace{\#41}_{\text{s-reg}}, R7 \quad ; \quad R7 \leftarrow 41 \text{ (0x0029)}$



$$\underbrace{4(R5)}_{\text{Indexed}}, \underbrace{6(R6)}_{\text{Indexed}}; \quad M[\underbrace{6+R6}_{EAd}] \leftarrow M[\underbrace{6+R6}_{EAd}] + M[\underbrace{4+R5}_{EAs}]$$

RS = Ox Coo b

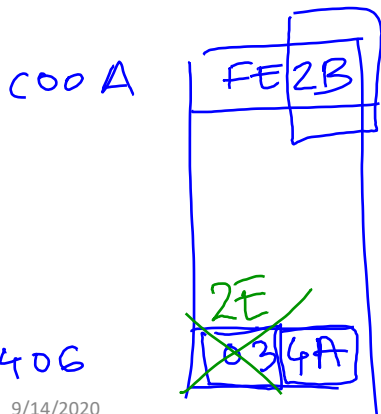
$$EA_S = 4 + RS = 4 + 0 \times 1006 = 0 \times 1006$$

$$EAX = 6 + R6 = 6 + 0x0401 = 0x0407$$

source operand: $M[EA_5] = M[\text{COOK}] = 0x2B$
 source index: $M[EA_{10}] = M[0407] = 0x03$

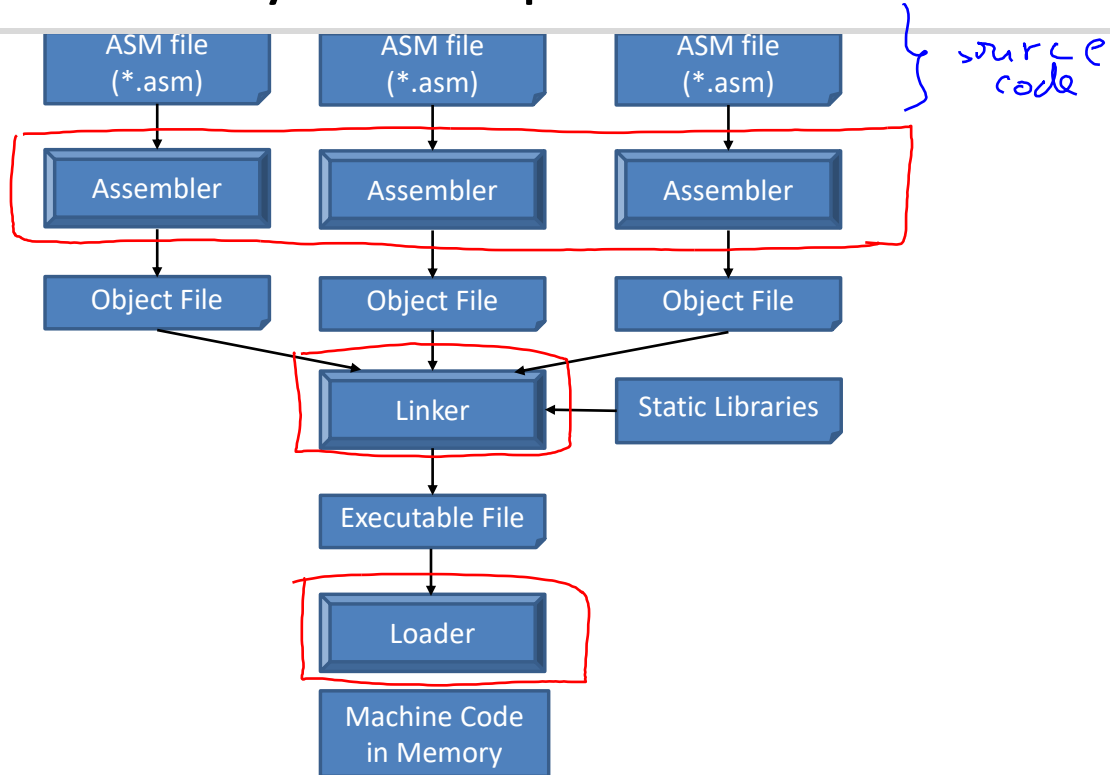
$$\begin{array}{r} 24 \text{ } 2B \\ + 01 \text{ } 03 \\ \hline 2F \end{array}$$

$$\begin{aligned} C &= 0 \\ V &= 0 \\ N &= 0 \\ Z &= 0 \end{aligned}$$



0×406

Assembly Development Flow



Assembly Language Directives

- Assembly language directives tell the assembler to
 - Set the data and program at particular addresses in address space
 - Allocate space for constants and variables
 - Define synonyms
 - Include additional files
 - ...
- Typical directives
 - Equate: assign a value to a symbol
 - Origin: set the current location pointer
 - Define space: allocate space in memory
 - Define constant: allocate space for and initialize constants
 - Include: loads another source file

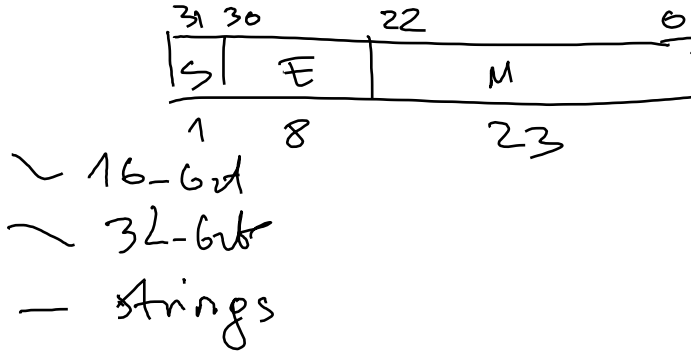
ASM Section Control Directives

| Description | ASM430 (CCS) | A430 (IAR) |
|--|---------------|------------|
| Reserve size bytes in the uninitialized sect. | <u>.bss</u> | - |
| → Assemble into the initialized data section | <u>.data</u> | RSEG const |
| Assemble into a named initialized data sect. | .sect | RSEG |
| → Assemble into the executable code | .text | RSEG code |
| Reserve space in a named (uninitialized) section | <u>.usect</u> | - |
| Align on byte boundary | .align 1 | - |
| Align on word boundary | .align 2 | EVEN |



Constant Initialization Directives

- .byte
- .float
- .word
- .long
- .string



Directives: Dealing with Constants

Hex

```

b1: .byte 5 ; allocates a byte in memory and initialize it with 5
b2: .byte -122 ; allocates a byte with constant -122
b3: .byte 10110111b ; binary value of a constant
b4: .byte 0xA0 ; hexadecimal value of a constant
b5: .byte 123q ; octal value of a constant
tf: .equ 25
    
```

synonym

octal

symbolic

5 3

mov.b b1, r7; r7 ← M[b1]
 117 r7 = 0x0005
 mov.b 0x3100, r7;

Table of symbols

| | |
|----|--------|
| b1 | 0x3100 |
| b2 | 0x3101 |
| b3 | 0x3102 |
| b4 | 0x3103 |
| b5 | 0x3104 |
| tf | 0x0019 |

0x3100

0x3102

0x3104

symbolic

mov.w b1, r8; r8 ← 8205

mov.w #b1, r9;

r9 ← 0x3100

mov.b b1+b3, r10; r10 ← 0x3103



Directives: Dealing with Constants

...
w1: .word 21 ; allocates a word constant in memory;
w2: .word -21
w3: .word tf
dw1: .long 100000 ; allocates a long word size constant in memory;
dw2: .long 0xFFFFFEA ; 100000 (0x0001_86A0)

Handwritten notes:
align 2
100,000 → 0001 86A0
upper / lower

6A 3106
w2 3108
w3 310A
dw1 310C
310E

| |
|-------|
| 00 15 |
| FF EB |
| 00 19 |
| 86 A0 |
| 0001 |
| FF EA |
| FFFF |

Table of symbols dw2 3110
3112
→ 3114

| |
|--------------|
| ... |
| w1 0x3106 |
| w2 0x3108 |
| w3 0x310A |
| dw1 0x310C |
| dw2 0x3110 |

21/16 = 1 15
1/16 = 0 1
21₁₀ = 15₁₆
1st compl: 00 15
FF EA
+ 1
FF EB

Directives: Dealing with Constants

s1: .byte 'A', 'B', 'C', 'D'; allocates 4 bytes in memory with string ABCD
s2: .byte "ABCD", '\0'; allocates 5 bytes in memory with string ABCD + NULL

• string "abcd" → 4 bytes
• cstring "abcd" → 5 bytes
TOS

| | |
|----|--------|
| | - - - |
| s1 | 0x3114 |
| s2 | 0x3118 |

| | | | |
|----|--------|-----|-----------|
| | | 'B' | '\0' |
| s1 | 0x3114 | 42 | 41 |
| | 0x3116 | 44 | 43 |
| s2 | 0x3118 | 42 | 41 |
| | 0x311A | 44 | 43 |
| | | | 00 |
| | | | ↑ NULL |

Table of Symbols

| Symbol | Value [hex] |
|--------|-------------|
| b1 | 0x3100 |
| b2 | 0x3101 |
| b3 | 0x3102 |
| b4 | 0x3103 |
| b5 | 0x3104 |
| tf | 0x0019 |
| w1 | 0x3106 |
| w2 | 0x3108 |
| w3 | 0x310A |
| dw1 | 0x310C |
| dw2 | 0x3110 |
| s1 | 0x3114 |
| s2 | 0x3118 |

Handwritten diagram showing a 4x4 grid structure. The grid is labeled with values on the left and top. The top labels are v_{26} and v_{16} . The left labels are 1100 , 1102 , 1104 , 1106 , 1108 , $110A$, and $110C$. There are additional labels $v_3 w$ and v_{46} to the left of the grid. The grid is divided into four 2x2 quadrants by a vertical and a horizontal line. A blue line outlines a 2x2 subgrid in the bottom half of the main grid, with a blue circle around the bottom-right cell. A blue line also outlines a 2x2 subgrid in the top half of the main grid, with a blue circle around the top-right cell. A blue line also outlines a 2x2 subgrid in the middle of the main grid, with a blue circle around the middle-right cell. A blue line also outlines a 2x2 subgrid in the bottom of the main grid, with a blue circle around the bottom-right cell.

V46

Diagram illustrating a 16-bit register structure. The register is divided into two main sections: a 4-bit field labeled "5:8" and a 12-bit field labeled "Memory[7:0]". Red handwritten annotations indicate that the "5:8" field is associated with variable `v26` and the "Memory[7:0]" field is associated with variable `v16`. A red box highlights the "Memory[7:0]" field, which contains a dash "--".

mov. b V16, r7 ;

Decimal/Integer Addition of 32-bit Numbers

- Write an assembly program that finds a sum of two 32-bit numbers
 - Input numbers are decimal numbers (8-digit in length)
 - Input numbers are signed integers in two's complement

E.g.:

lint1: .long 0x45678923

lint2: .long 0x23456789

Binary
Add

68AC FOAC

Decimal
addition

$$\begin{array}{r}
 0x45678923 \\
 + 0x23456789 \\
 \hline
 69135712
 \end{array}$$

lint1:

lint2:

sumi:

sumd:

| |
|------|
| 8923 |
| 4567 |
| 6789 |
| 2345 |
| : |
| : |
| FOAC |
| 68AC |
| : |
| : |
| 5712 |
| 6913 |

Allocate Space & Start Program

| | | |
|-------------------------------------|---|-------------------------------------|
| clc | : | <u>bic #1, R2</u> |
| mov.w <u>lint1</u> , <u>(R8)</u> | : | $R8 \leftarrow M[lint1]$ |
| dadd.w <u>lint2</u> , <u>R8</u> | : | $R8 \leftarrow R8 + M[lint2] + C$ |
| { mov.w <u>R8</u> , <u>lsund</u> | : | $M[lsund] \leftarrow R8$ |
| | : | |
| mov.w <u>lint1+2</u> , <u>R8</u> | : | $R8 \leftarrow M[lint1+2]$ |
| dadd.w <u>lint2+2</u> , <u>R8</u> | : | $R8 \leftarrow R8 + M[lint2+2] + C$ |
| mov.w <u>R8</u> , <u>lsund+2</u> | : | $M[lsund+2] \leftarrow R8$ |

Main Code (Ver. 1)

```

mov.w    lnt1, R8
→ add.w  lnt2, R8    ;   R8 ← R8 + M[lnt2]
{
  mov.w   R8, lsumi
  mov.w   lnt1+2, R8
}
→ addc.w lnt2+2, R8
  mov.w   R8, lsumi+2
  
```

Main Code (Ver. 2)

```
mov.w #Lint1, R4
mov.w #lsmd, R8
mov.w #2, R5
```

```
clr R10
```

lda:

```
mov.w 4(R4), R7
```

```
mov.w R10, R2
```

```
→ cladd.w @R4+, R7
```

```
mov.w R2, R10
```

```
mov.w R7, 0(R8)
```

```
→ dec.w R5
```

```
jnz lda
```

R5 is step counter

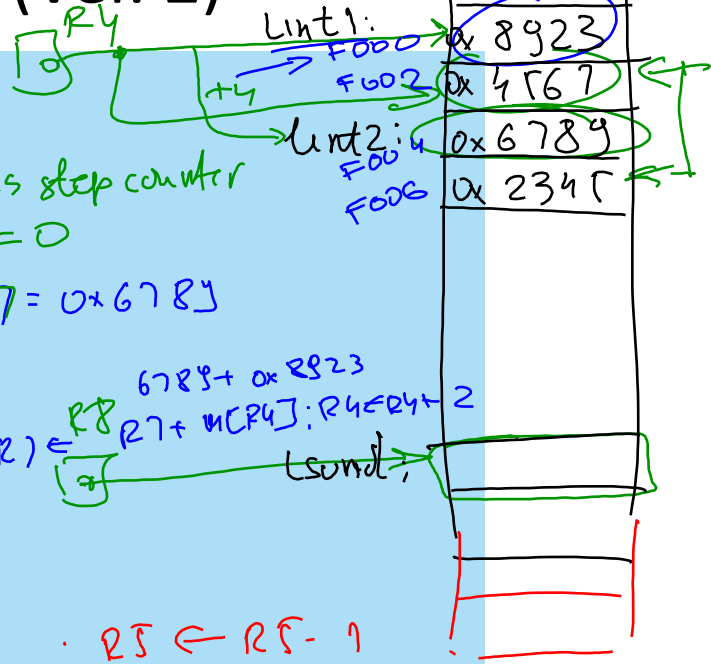
R10 = 0

R7 = 0x6789

?

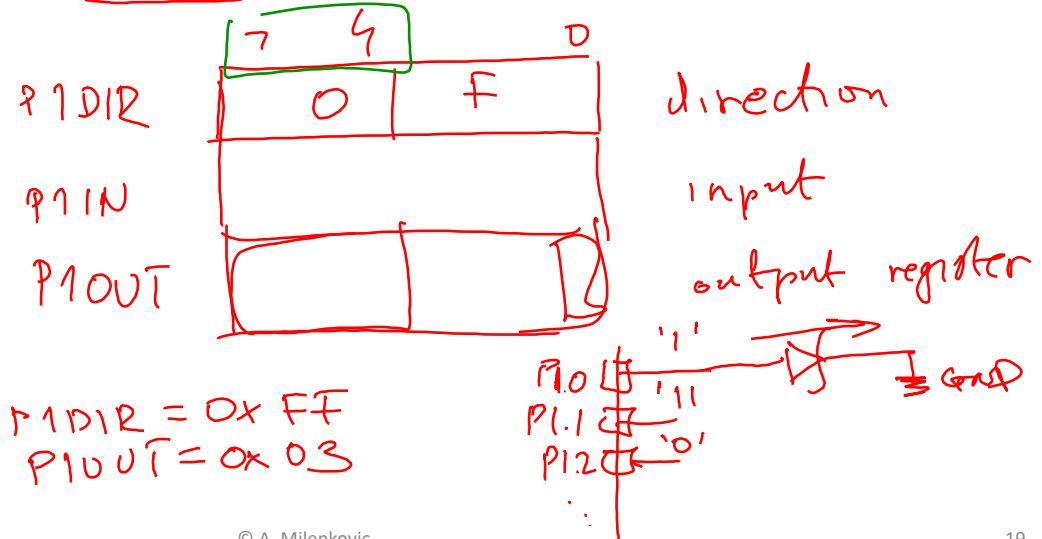
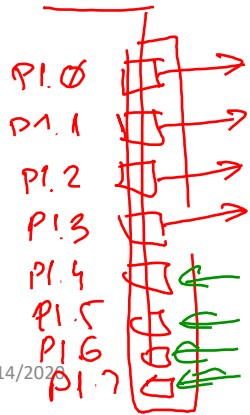
6789 + 0x8923
R7 ← R7 + M[R4]; R4 ← R4 + 2

R5 ← R5 - 1



Count Characters 'E' in a String

- Write an assembly program that processes an input string to find the number of characters 'E' in the string
- The number of characters is "displayed" on the port 1 of the MSP430



Count Characters 'E' in a String

```

;-----
; File      : Lab4_D1.asm (CPE 325 Lab4 Demo code)
; Function   : Counts the number of characters E in a given string
; Description: Program traverses an input array of characters
;            : to detect a character 'E'; exits when a NULL is detected
; Input      : The input string is specified in myStr
; Output     : The port P1OUT displays the number of E's in the string
; Author     : A. Milenkovic, milenkovic@computer.org
; Date      : August 14, 2008
;-----

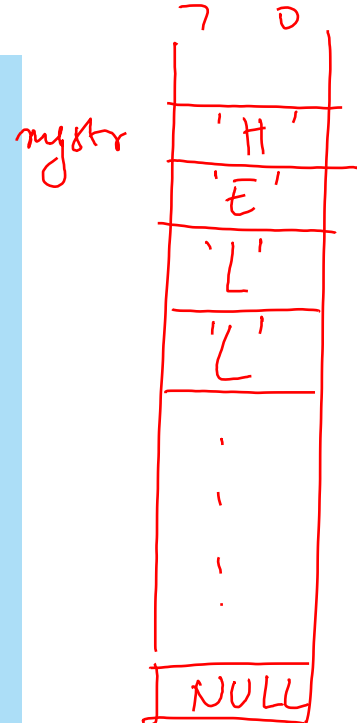
.cdecls C,LIST,"msp430.h"      ; Include device header file

;-----
.def      RESET                ; Export program entry-point to
                                ; make it known to linker.
myStr:    .string "HELLO WORLD, I AM THE MSP430!"
;-----
.text                                     ; Assemble into program memory.
.retain                                     ; Override ELF conditional linking
                                           ; and retain current section.
.retainrefs                             ; And retain any sections that have
                                           ; references to current section.

;-----
RESET:    mov.w    # STACK_END,SP        ; Initialize stack pointer
          mov.w    #WDTPW|WDTHOLD,&WDTCTL ; Stop watchdog timer
    
```

Handwritten notes on the code:

- myStr: (circled in red)
- RESET (circled in green)
- myStr: (circled in red)
- RESET: (circled in green)
- mov.w # STACK_END,SP (circled in green)
- mov.w #WDTPW|WDTHOLD,&WDTCTL (circled in green)
- myStr: (circled in red)
- RESET: (circled in green)
- mov.w # STACK_END,SP (circled in green)
- mov.w #WDTPW|WDTHOLD,&WDTCTL (circled in green)



Count Characters 'E' in a String

```
main:
```

mov.w #mystr, R4

CLR. 6 25

counts 't's

$$\cdot 26 \in \mathbb{N}[R_4]; 24 \in R_4 + 1$$

1. $WVLL = \alpha \times \underline{00}$

gnext:

~~comp.b~~ ~~#0, 26~~

jeq lehu

$$0 \text{ cmp.b } \#t, r6$$

jne, jnext

in c.w RS

imp qnext

end: mov.b RS, &P10UT

```
; Stack Pointer definition
```

```
.global __STACK_END
.sect .stack
```

```
; Interrupt Vectors
```

```
.sect ".reset"  
.short RESET  
.end
```

```
; MSP430 RESET Vector
```

6 is # LPM4, R2

15 87 0 1

| | |
|-----|-----|
| 'E' | 'H' |
| 'L' | 'L' |

24 mystr:

The diagram shows a blue box containing the letter 'a'. A blue line extends from the box and curves upwards and to the right, ending in a red arrowhead. A red line also extends from the box, curving downwards and to the right, ending in a red arrowhead. The text '24 mystr:' is written above the box.

| | | |
|--|------|---|
| | 7 | 0 |
| | (1) | |
| | E' | |
| | L' | |
| | : | |
| | | |
| | | |
| | | |
| | NVLL | |