ASSEMBLY LANGUAGE!!!!!!!!!!

## Five Instructions

mov.w src, dst

add.w src, dst

rra.w dst

jmp label

nop

## First Task

Find the average value of the set of numbers {2, -43, 7, 19}

/\*

\* Find the average value of the set of numbers {2, -43, 7, 19}

\*/

mov.w #\_\_STACK\_END,SP

mov.w #WDTPW|WDTHOLD,&WDTCTL

mov.b #2, R4 ; R4 <- 2

add.b #-43, R4 ; R4 <- R4 + -43

add.b #7, R4 ; R4 <- R4 + 7

add.b #19, R4 ; R4 <- R4 + 19

rra.b R4

rra.b R4

jmp main

nop

## Assembler Directives

- Assemble code and data into specified sections

.data ; everything after this goes to RAM

.text ; everything after this goes to FRAM

- Reserve space in memory (initialized to zero)

.space 6 ; reserve 6 bytes of space

- Initialize memory to desired values

.word 0xB, 0xC ; initialize words

.byte -1, 5, 3 ; initialize bytes (Don’t use odd numbers like in example, big no-no)

- Define global variables

array: .word 0x1, 0x2, 0x3, 0x4

- Define symbolic constants – no memory reserved

scon: .set 4

## The Program Counter R0/PC

- The core register R0 is the Program Counter PC

- The program counter points to the next instruction to be executed

When we look at the PC register, we see the address of the next instruction

## Second Task

- Define word variables x=5 and y=8 in RAM and reserve space for word variable sum

.data

x: .word 5 x = 0x1C00

y: .word 8 y = 0x1C02

sum: .space 2 sum = 0x1C04

mov.w x, sum

add.w y, sum

## Arrays

Arrays aren’t real!!!!!!!!!!!!

But lying is, so we’ll emulate them with assembler directives and labels

array1: .word 0x0100, 0x0200, 0x0300 // array1, array1+2, array1+4

array2: .byte 0x01, 0x02, 0x03 // array2, array2+1, array2+2

## Indexed Mode of Addressing

Syntax of indexed mode

array1: .word 0x0100, 0x0200, 0x0300

mov.w array1(R4), R5 ; syntax is x(R)

; x is the array name

; R is a core register

; copies from x + (@R)

mov.w #2, R4

mov.w array1(R4), R5

Be careful when working with byte vs word arrays

## Indexed Mode and Byte Arrays

array2: .byte 0x10, 0x20, 0x30

mov.b &array2, R5

mov.b &array2+1, R5

mov.b &array2+2, R5

Same as:

mov.w #0, R4 ; R4 = 0 will be the index

mov.b array2(R4), R5 ; R5 = array2[R4]

inc.w R4 ; R4++

add.b array2(R4), R5 ; R5 += array2[R4]

inc.w R4 ; R4++

add.b array2(R4), R5 ; R5 += array2[R4]

## Indexed Mode and Word Arrays

array1: .word 0x0100, 0x0200, 0x0300

mov.w &array1, R5

add.w &array1+2, R5

add.w &array1+4, R5

## More Instructions

MSP430 has many emulated instructions

woop woop