CSSE2010/CSSE7201
Learning Lab 10

Microchip Studio Introduction

http://responsewaresg.net

Session ID: CSSE2010EXT

School of Information Technology and Electrical Engineering
The University of Queensland

Today

- AVR Assembly Language✓
  - Instruction Intro
    - Documentation
  - Machine Code Equivalent
- Microchip Studio (previously known as Atmel Studio)
  - Assembling and Simulating code
  - Objectives: you should know how to create a project for the ATmega324A/ATmega328, enter assembly language instructions, build the project, simulate instructions and examine register values

AVR Documentation

- ATmega324A/ATmega328P Datasheet ✓
- AVR Instruction Set Manual
- AVR Instruction Set Reference

- Can be found on Blackboard

ADD Instruction – status bits

Status Register (SREG) and Boolean Formula:

Add instruction cannot modify these

		Sign corrected	Overflow	Negative	Zero	Carry ✓
I	T 2	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

S: $N \oplus V$, For signed tests. ✓

V: $Rd7\bullet Rr7\bullet\overline{R7}+\overline{Rd7}\bullet\overline{Rr7}\bullet R7$

Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7 ✓

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7}\bullet\overline{R6}\bullet\overline{R5}\bullet\overline{R4}\bullet\overline{R3}\bullet\overline{R2}\bullet\overline{R1}\bullet\overline{R0}$

Set if the result is \$00; cleared otherwise.

C: $Rd7\bullet Rr7+Rr7\bullet\overline{R7}+\overline{R7}\bullet Rd7$ ✓

Set if there was carry from the MSB of the result; cleared otherwise. ✓

R (Result) equals Rd after the operation.

```
 0 1
   0111
+  0110
   ----
   1101
```

C: 0   V: 1

Z: 0   S: $1 \oplus 1 = 0$

N: 1

What is the result of this program?

```
ldi r16, 0x56
ldi r17, 0x34
add r17, r16
```

hex

A. r17 will contain 0x90
r16 will contain 0x56

B. r17 will contain 0x34
r16 will contain 0x90

[✓ C.] r17 will contain 0x8A ✓
r16 will contain 0x56

D. r17 will contain 0x34
r16 will contain 0x8A

r16 ← 0x56

r17 ← 0x34

r17 ← 0x34 + 0x56

0x8A

Constants in AVR Assembly Language

- Don’t have to be hexadecimal
- AVR examples (all mean the same)
  - `ldi r17, 0x34`
  - `ldi r17, $34` hex
  - `ldi r17, 52` decimal
  - `ldi r17, 064` octal
  - `ldi r17, 0b00110100` binary

0101 101_8

Microchip Studio

- Create Assembler Project
  - Start as per Microchip Studio Tutorial
- Enter this program:

```
ldi r16, 0x56
ldi r17, 0x34
add r17, r16
```

- Step through it, check register values after each step

6 - 8 -> -ve

8 - 8 -> 0 -> Z:1

C:1 V:0 Z:0

N:0

Compare Instructions

r5 0101
r6 - 0100
0001

-> 0101
+ 1011
0001

- **Compare instructions**
  - Compare two registers, e.g.
    `cp r5, r6` ✓
    - ALU does r5-r6, discards result but adjusts status register flags appropriately ✓
  - Compare register value with constant, e.g.
    `cpi r17, 79`
    - ALU does r17 - 79, discards result but adjusts status register flags appropriately
    - **NOTE**: like **`ldi`**, **`cpi`** only works with r16...r31 ✓
- There is also a **tst** instruction, e.g. **`tst r18`**, for examining one register
  - **`tst rd`** is the same as **`and rd, rd`**

What value will be in the status register after these instructions?

```
→ ser r17
→ ldi r18, 0xF0
→ cp r17, r18
```

30%	1.	Z=0	N=0	V=0	✓
20%	2.	Z=0	N=0	V=1	
30%	3.	Z=0	N=1	V=0	
	4.	Z=0	N=1	V=1	
20%	5.	Z=1	N=0	V=0	
	6.	Z=1	N=0	V=1	
	7.	Z=1	N=1	V=0	
	8.	Z=1	N=1	V=1	

Z=0 V=0

N=0

r17 ← 0b1111 1111

r18 ← 0xF0

0b1111 0000

```
  1111 1111
− 1111 0000
-----------

  1111 1111
  1111 1111
+ 0000 1111
          1
-----------
  0000 1111
```

Bit masking

- Sometimes interested in only a few bits of a register
- Can use a bitwise AND to *mask* out bits not of interest
- Note: Status register flags will be set also
- Example:

		↓	↓			↓	↓	↓
8-bit #	a	b	c	d	e	f	g	h
Mask	0	1	1	0	0	1	1	1
Bitwise AND	0	b	c	0	0	f	g	h

Exercises

S V N Z

0 0 0 0

```
 01010101
 00010111
 --------
 00010101
```

0x15

- Write an AVR assembly language program which performs the following operations on values 0x55 and 23_{10} and puts the result in the given register
  - Bitwise AND → r16
  - Bitwise OR → r5
  - Bitwise EOR → r17
  - Addition → r6
  - Difference (subtraction) → r7
  - One’s complement (inversion) of 0x55 → r8
  - Two’s complement (negation) of 0x55 → r9

```
ldi r16, 0x55
ldi r17, 23

and r16, r17
```

```
ldi r16, 0x55
andi r16, 23
```

- Predict the results AND status register values (Z,C,N,V) after each operation
- Simulate the code in Atmel Studio and check your predictions
- Write AVR assembly language code snippets to
  - ✓ Copy the least significant three bits of register r7 to register r8. Other bits of r8 should be 0.
  - Toggle the most significant four bits of register r9. Other bits should be unchanged.
  - Clear the least significant four bits of register r10. Other bits should be unchanged.
  - Set the most significant four bits of register r11 to be 1. Other bits should be unchanged.
  - Test these code snippets

MASK: 0000111

```
ldi r16, 0x55
mov r7, r16
mov r8, r7
ldi r16, 0b00000111
and r8, r16 -> r16
 = 0x05
```