CSSE2010

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/ATmega328 Datasheet
- AVR Instruction Set Manual
- AVR Instruction Set Reference

- Can be found on Blackboard

ADD Instruction – status bits

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

S: N ⊕ V, For signed tests.

V: Rd7•Rr7•$\overline{R7}$+$\overline{Rd7}$•$\overline{Rr7}$•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: $\overline{R7}$• $\overline{R6}$ •$\overline{R5}$• $\overline{R4}$ •$\overline{R3}$ •$\overline{R2}$ •$\overline{R1}$ •$\overline{R0}$
Set if the result is $00; cleared otherwise.

C: Rd7 •Rr7 +Rr7 •$\overline{R7}$+ $\overline{R7}$ •Rd7
Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

```
  1 1 1 1 1
+ 0 1 1 0
-----------
  0 1 0 1
```

C:1 Z:0 N:0 V:0

S:0

What is the result of this program?

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

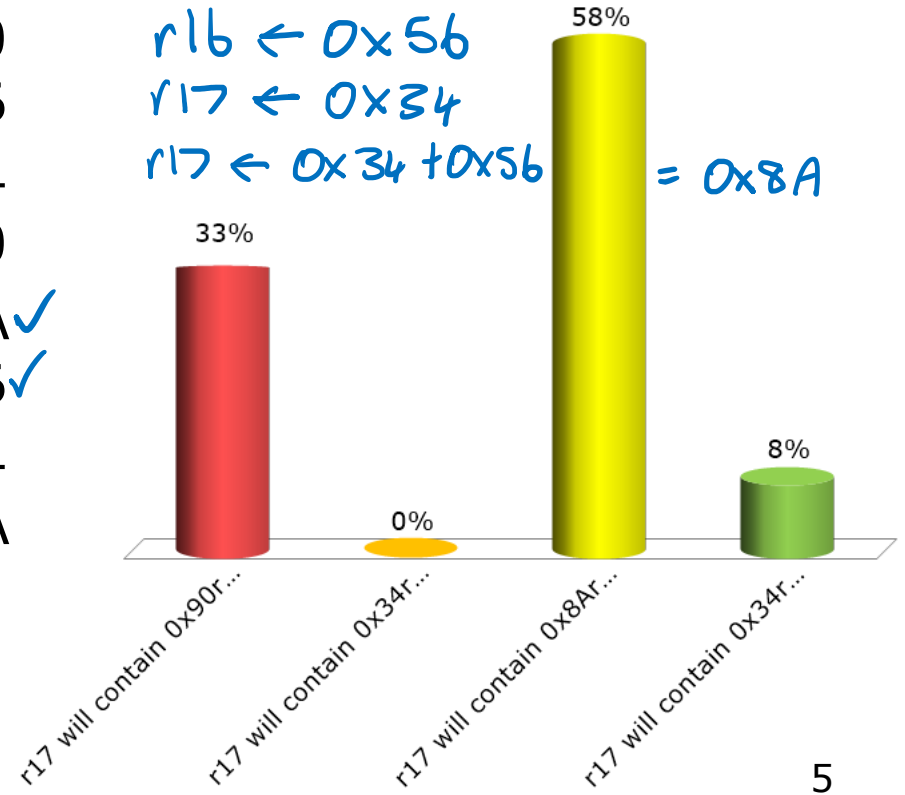
A. r17 will contain 0x90
   r16 will contain 0x56

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



http://responsewaresg.net

Session ID: CSSE2010EXT

Constants in AVR Assembly Language

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

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

r5 - r6 -> +ve -> r5 > r6
r5 - r6 -> -ve -> r6 > r5

Compare Instructions

r5 - r6 -> z:1 -> r5 == r6

- **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`**

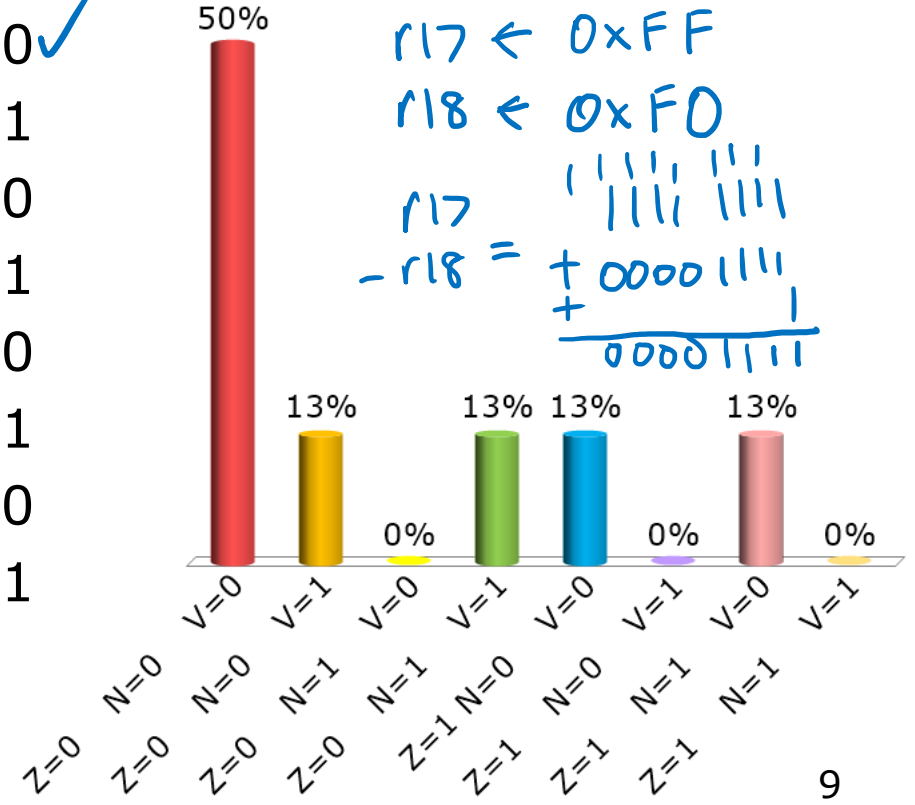
What value will be in the status register after these instructions?

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

Z:0 N:0 V:0

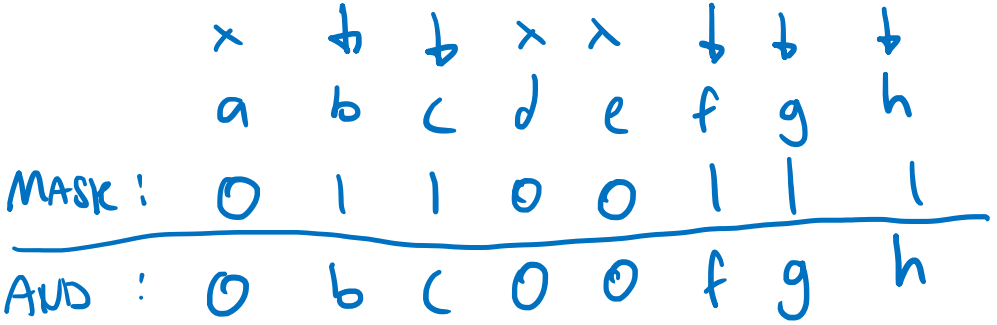
1. Z=0 N=0 V=0 ✓
2. Z=0 N=0 V=1
3. Z=0 N=1 V=0
4. Z=0 N=1 V=1
5. Z=1 N=0 V=0
6. Z=1 N=0 V=1
7. Z=1 N=1 V=0
8. Z=1 N=1 V=1

r17 ← 0xFF
r18 ← 0xF0
r17 = 11111111
-r18 = +00001111
+ 1
00001111



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:



Interested in b, c, f, g, h

Exercises

0x55 = 01010101
00000111
00000101
= 0x05

- 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
or  r16, r17
mov r5,  r16
```

```
01010101
00010111
--------
01010111 -> 0x57
```

Z:0 C:-
N:0 V:0

```
ldi r16, 0x55
mov r7,  r16
mov r8,  r7
```

- 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

```
ldi r16, 0b00000111
and r8,  r16
```