

Microchip Studio Introduction

htp://responsewaresg.net

Session ID: CSSEZOIDEXT

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

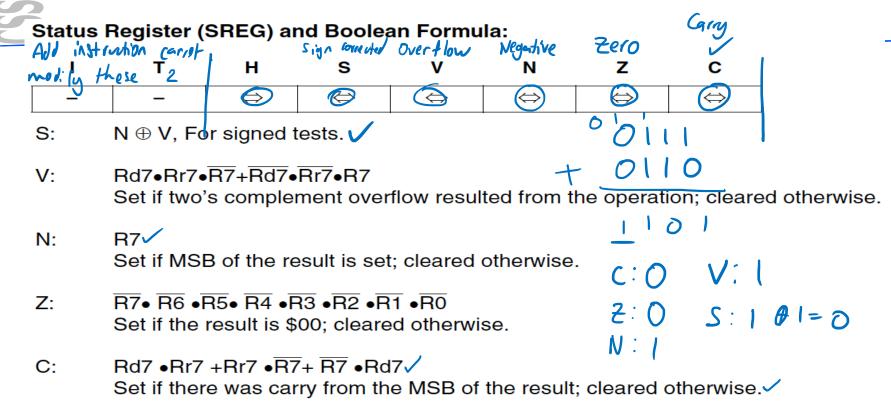


AVR Documentation

- ATmega324A/ATmega328 Datasheet
- AVR Instruction Set Manual
- AVR Instruction Set Reference

Can be found on Blackboard

ADD Instruction – status bits



R (Result) equals Rd after the operation.



What is the result of this program?

Idi r16, 0x56
 Idi r17, 0x34
 add r17, r16

- A. r17 will contain 0x90
 r16 will contain 0x56
 - B. r17 will contain 0x34 r16 will contain 0x90
 - r17 will contain 0x8A r16 will contain 0x56
 - D. r17 will contain 0x34 r16 will contain 0x8A

$$116 \leftarrow 0 \times 56$$

$$117 \leftarrow 0 \times 34$$

$$117 \leftarrow 0 \times 34 + 0 \times 56$$

$$0 \times 84$$



Constants in AVR Assembly Language

Don't have to be hexadecimal

ldi r17, 0b00110100

AVR examples (all mean the same)

```
Idi r17, 0x347 hex
Idi r17, $34
Idi r17, 52 deim
Idi r17, 064 octal
```

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

8-8->-ve 8-8->-> 2:1 C:1 V:0 2:0 *Compare Instructions N:0

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?

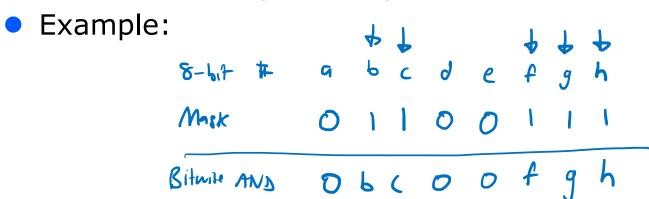
→ ser r17 \rightarrow ldi r18, 0xF0 \rightarrow cp r17, r18

30%	1. Z=0	N=0 $V=0$	
20%	2. Z=0	N=0 $V=1$	118 < OXFD
50%	3. $Z=0$	N=1 $V=0$	06/11/0000
	4. $Z=0$	N=1 $V=1$	
20%	5. $Z=1$	N=0 $V=0$	(1111111
	6. $Z=1$	N=0 $V=1$	- ((() () () ()
	7. $Z=1$	N=1 $V=0$	(11/1/11
	8. $Z=1$	N=1 $V=1$	
		7=0 V=0	+ 00001111
		N = 0	00001111
			9



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





Exercises OOC

r16, r17



- Bitwise AND \rightarrow r16
 Bitwise OR \rightarrow r5
 - Bitwise EOR → r17
 - Addition → r6
 - Difference (subtraction) → r7
 - One's complement (inversion) of 0x55 → r8
 - Two's complement (negation) of $0x55 \rightarrow r9$ Predict the results AND status register values (Z,C,N,V) after each operation $\nearrow N$
- 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

00010111 00010101 0×15 as5 and

01010101

MAIK: 0000 111

mov 17, 16

rs, r7

-> 116 - 0x05

11