ECE375
Introduction to AVR Simulation with Atmel Studio

TA: Han Jang
School of Electrical Engineering and Computer Science
Oregon State University
Announcement

- Check-off to your lab TA only.

- You must include Source code and Challenge code in your lab report.

- If you don’t submit either lab reports or code files, or don’t do the demo, you will get 0 credit for the lab.

- Don’t forget to write comments!
  - Your AVR assembly code need to be well-commented.
AVR Simulation

- Learn how to use the Atmel Studio simulator
- Set Break Points
- Start Debugging and Break
- Open Processor, I/O port, and Memory Window
Break Points

```asm
; INIT: ; The initialization routine
ldi mpr, low(RAMEND) ; initialize Stack Pointer
cut SPL, mpr
cut mpr, high(RAMEND)
cut SPH, mpr

; MAIN:
clr r0 ; *** SET BREAKPOINT HERE *** (#1)
dec r0 ; initialize r0 value

clr r1 ; *** SET BREAKPOINT HERE *** (#2)
ldi i, $04
Loop: 1sl r1 ; initialize r1 value
inc r1
1sl r1
dec i
brne Loop ; *** SET BREAKPOINT HERE *** (#3)

clr r2 ; *** SET BREAKPOINT HERE *** (#4)
ldi i, $0F
Loop2: inc r2 ; initialize r2 value
cp r2, i
brne Loop2 ; *** SET BREAKPOINT HERE *** (#5)

mov r3, r2 ; *** SET BREAKPOINT HERE *** (#6)

; Note: At this point, you need to enter several values
; directly into the Data Memory. FUNCTION is written to
; expect memory locations $0101:$0100 and $0103:$0102
; to represent two 16-bit operands.
;
; So at this point, the contents of r0, r1, r2, and r3
; MUST be manually typed into Data Memory locations
```
Start debugging and break
Simulator

- Main.asm file
- Simulator selected as the tool for debugging
- Programming settings: Erase entire chip, Preserve EEPROM
- Select Stimuli File for Simulator with instructions to activate stimuli in breakmode from menu Debug->Execute Stimulifile, then continue execution
### Processor Status

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Counter</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Stack Pointer</td>
<td>0x0000</td>
</tr>
<tr>
<td>X Register</td>
<td>0x0000</td>
</tr>
<tr>
<td>Y Register</td>
<td>0x0000</td>
</tr>
<tr>
<td>Z Register</td>
<td>0x0000</td>
</tr>
<tr>
<td>Status Register</td>
<td>[unknown]</td>
</tr>
<tr>
<td>Cycle Counter</td>
<td>0</td>
</tr>
<tr>
<td>Frequency</td>
<td>1.000 MHz</td>
</tr>
<tr>
<td>Stop Watch</td>
<td>0.00 µs</td>
</tr>
</tbody>
</table>

#### Registers

- R00: 0x00
- R01: 0x00
- R02: 0x00
- R03: 0x00
- R04: 0x00
- R05: 0x00
- R06: 0x00
- R07: 0x00
- R08: 0x00
- R09: 0x00
- R10: 0x00
- R11: 0x00
- R12: 0x00
- R13: 0x00
- R14: 0x00
- R15: 0x00
- R16: 0x00
- R17: 0x00
I/O Ports
Memory
Questions?