ECE375
Introduction to AVR Simulation with Microchip Studio

TA:
School of Electrical Engineering and Computer Science
Oregon State University
Supported Architecture

- Only **Windows** support the Microchip Studio
- Microchip Studio currently does not support a simulator for ATmega32U4. Select the ATmega128 device instead.
Goal of this Pre Lab

- Learn how to use the Microchip Studio simulator
- Set Break Points
- Start Debugging and Break
- Observe data using Processor, I/O port, and Memory window
Break Points

```assembly
;********************
INIT:
  ldi  mpr, low(RAMEND) ; The initialization routine
  cut  SPL, mpr
  ldi  mpr, high(RAMEND)
  cut  SPH, mpr

;********************
;*  Main Program  *
;********************
MAIN:
  clr  r0        ; *** SET BREAKPOINT HERE *** (#1)
  dec  r0        ; Initialize r0 value

  clr  r1        ; *** SET BREAKPOINT HERE *** (#2)
  ldi  i, $04
  loop:          ; Initialize r1 value
    inc  r1
    inc  r1
    dec  i
  brne loop     ; *** SET BREAKPOINT HERE *** (#3)

  clr  r2        ; *** SET BREAKPOINT HERE *** (#4)
  ldi  i, $0F
  loop2:         ; Initialize r2 value
    inc  r2
  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

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 $0100$, $0101$, $0102$, and $0103$ respectively.
Simulator

Lab3 - AtmelStudio

Build
Build Events
Toolchain
Device
Tool
Components
Advanced

Selected debugger/programmer

Tool
Simulator

Programming settings

- Erase entire chip
- Preserve EEPROM

Select Stimuli File for Simulator

Stimuli File
Activate stimuli when 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>[values]</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

- Analog Comparator (AC)
- Analog-to-Digital Converter
- Bootloader (BOOT_LOAD)
- CPU Registers (CPU)
- EEPROM (EEPROM)
- External Interrupts (EXINT)
- I/O Port (PORTA)
- I/O Port (PORTB)
- I/O Port (PORTC)
- I/O Port (PORTD)
- I/O Port (PORTE)
- I/O Port (PORTF)
- I/O Port (PORTG)
- JTAG Interface (JTAG)
- Other Registers (MISC)
- Serial Peripheral Interface (SPI)

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Value</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>PINA</td>
<td>0x39</td>
<td>0xFF</td>
<td></td>
</tr>
<tr>
<td>DDRA</td>
<td>0x3A</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>PORTA</td>
<td>0x3B</td>
<td>0x00</td>
<td></td>
</tr>
</tbody>
</table>
Memory
Announcements

- There is a tutorial video for this assignment.
Questions?