ENGR-E 399/599: Embedded systems reverse engineering

Lecture 6: AVR architecture and ISA

Austin Roach
ahroach@iu.edu

February 17, 2022

Mystery function

```
   0x00000000   01c040e2   sub ip, r0, 1
┌─> 0x00000004   0120fce5   ldrb r2, [ip, 1]!
|  0x00000008   0130d1e4   ldrb r3, [r1], 1
|  0x0000000c   030052e0   subs r0, r2, r3
|  0x00000010   1eff2f11   bxne lr
|  0x00000014   000052e3   cmp r2, 0
└─< 0x00000018   f9ffff1a   bne 4
   0x0000001c   1eff2fe1   bx lr
```

- How many arguments does the function take?
- What are the types of the arguments?
- What does the function do?
- What does the function return?

Mystery function revealed

```
int strcmp(const char *s1, const char *s2)
```

Description

The `strcmp()` function compares the two strings `s1` and `s2`. It returns an integer less than, equal to, or greater than zero if `s1` is found, respectively, to be less than, to match, or be greater than `s2`.

Return value

The `strcmp()` function returns an integer less than, equal to, or greater than zero if `s1` is found, respectively, to be less than, to match, or be greater than `s2`.

Outline

- AVR overview
  - Registers
  - Memory architecture
  - Reset and interrupts
  - Quirks
- Demo
  - Hardware and disassembly environment
  - Disassembly of simple program
- Assignment 3 introduction

AVR history

- 8-bit microcontroller architecture
- Developed in the mid 1990s
- Controlled by Atmel for almost its entire history

General-purpose registers

- 32 8-bit general purpose registers
  - `R0`–`R31`
- 6 registers can be used as 3 pairs of 16-bit address registers
  - `X`, `Y`, and `Z`
- 8 registers can be used as 4 pairs for 16-bit integer arithmetic
  - `X`, `Y`, `Z`, and `R25:R24`, sometimes called '`W`'

R24	Wlo	W
R25	Whi	
R26	Xlo	X
R27	Xhi	
R28	Ylo	Y
R29	Yhi	
R30	Zlo	Z
R31	Zhi	

Other registers

- Program counter (PC)
- Stack pointer (SP)
- Status register (SREG)
  - C (Carry), Z (Zero), N (Negative), V (Twos-complement overflow), S (N$\oplus$V for signed tests), H (Half-carry flag), T (Transfer bit), and I (Global interrupt enable/disable flag)
- I/O registers

Reference

ATmega 328p datasheet, Section 11: AVR CPU Core

Flags aren't just for comparisons

16-bit math

```
uint16_t add_16bits(uint16_t a,
                    uint16_t b)
{
    return a + b;
}
```

Compiles to:

```
add r24, r22
adc r25, r23
ret
```

32-bit math

```
uint32_t add_32bits(uint32_t a,
                    uint32_t b)
{
    return a + b;
}
```

Compiles to:

```
add r22, r18
adc r23, r19
adc r24, r20
adc r25, r21
ret
```

Memory architecture

- Modified Harvard architecture
  - Separate code ("flash"), data ("SRAM") memory
- General-purpose registers, I/O registers memory mapped to data memory address space
- Data EEPROM accessible through I/O control registers

Flash program memory

- 16 bits in width
  - Program counter holds a *word* address
- `LPM` - Load Program Memory
- `SPM` - Store Program Memory
  - Load/store program memory operates (effectively) on a *byte* address

Reference

ATmega 328p datasheet, Section 12.2: In-system reprogrammable flash program memory

Data memory (SRAM)

- 8 bits in width
- `LD`, `LDS`, `LDD` - Load from data memory
- `ST`, `STS`, `STD` - Store to data memory

- `0x0-0x1f`: Mapped general-purpose registers
- `0x20-0x5f`: I/O registers
- `0x60-0xff`: Extended I/O registers
- `0x100-0x8ff`: Internal SRAM

Reference

ATmega 328p datasheet, Section 12.3: SRAM data memory

EEPROM data memory

- Controlled through I/O registers
- Unused for our assignments

Reference

ATmega 328p datasheet, Section 12.4: EEPROM data memory

I/O control registers

- Provide a mechanism to configure and control peripheral I/O blocks
- Will be covered in more depth next lecture
  - For this week: loading from `UDR0` loads a byte received over the UART

I/O access: data-memory-mapped

- Data memory address space from 0x20-0xff
- Accessed with memory load/store instructions
- All I/O registers can be accessed this way

Reference

ATmega 328p datasheet, Section 35: Register summary

I/O access: I/O specific instructions

- Access with `IN`, `OUT`, `SBI`, `CBI`, `SBIC`, and `SBIS` instructions
  - Only the first 64 I/O registers (0-0x3f) are accessible by the `IN` and `OUT` instructions
  - Only the first 32 I/O registers (0-0x1f) are accessible by the `SBI`, `CBI`, `SBIC`, and `SBIS` instructions
- *I/O register number is data memory address minus 0x20*

Reference

ATmega 328p datasheet, Section 12.5: I/O memory

Control flow – unconditional

- `RJMP`, `JMP`
  - Unconditional jumps
- `RCALL`, `CALL`
  - Call a subroutine
  - Push return address onto stack
  - Return with `RET`

Control flow – conditional branches

- `BRBC bit,displacement`
  - Branch if bit in `SREG` is cleared
  - Synonym for various `BR*` instructions
  - Ghidra has the operands backwards
- `BRBS bit,displacement`
  - Branch if bit in `SREG` is set
  - Synonym for various `BR*` instructions
  - Ghidra has the operands backwards

Control flow – conditional branch synonyms

BRBS Cflg	BRCS	BRLO
BRBC Cflg	BRCC	BRSH
BRBS Zflg	BREQ	
BRBC Zflg	BRNE	
BRBS Nflg	BRMI	
BRBC Nflg	BRPL	
BRBS Vflg	BRVS	
BRBC Vflg	BRVC	
BRBS Sflg	BRLT	
BRBC Sflg	BRGE	

BRBS Hflg	BRHS
BRBC Hflg	BRHC
BRBS Tflg	BRTS
BRBC Tflg	BRTC
BRBS Iflg	BRIE
BRBC Iflg	BRID

Control flow – skip instructions

- `CPSE Rd,Rr`
  - Skip next instruction if register contents are equal
- `SBRC Rr,b`
  - Skip next instruction if bit in register is cleared
- `SBRS Rr,b`
  - Skip next instruction if bit in register is set
- `SBIC IO5,b`
  - Skip next instruction if bit in I/O register is cleared
- `SBIS IO5,b`
  - Skip next instruction if bit in I/O register is set

These instructions can be confusing. `SkipAVR8Fixup.java` can help.

Control flow – skip instruction examples

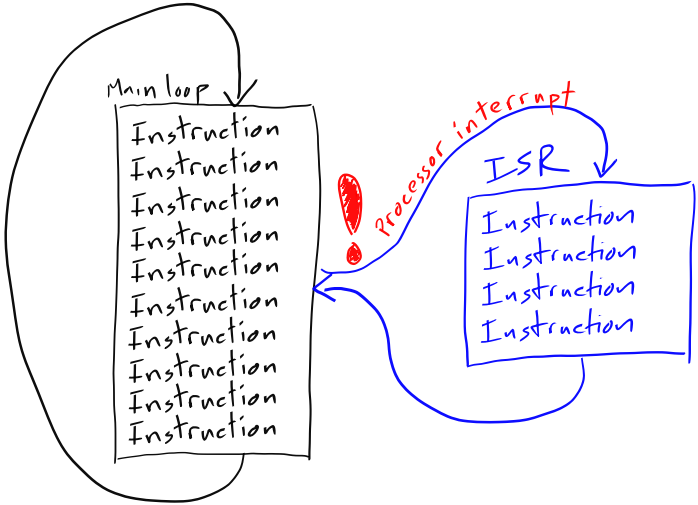
```
...
cpse R21,R1
rjmp 0x31c
call 0x152
...

...
cpse R18,Wlo
ldi R21,1
subi R20,0xff
...
```

Reset and interrupt vectors

- Relocatable based on fuse settings
- Our table is at program memory address 0

Interrupts


Main loop
Instruction
Instruction
Instruction
Instruction
Instruction
Instruction
Instruction
Instruction
Instruction
Instruction
Processor interrupt
ISR
Instruction
Instruction
Instruction
Instruction

Enabling interrupts

Global interrupts:

- Global interrupt (`I`) flag in `SREG`
- `SEI` (enable) and `CLI` (disable) instructions

Enabling specific interrupts:

- Enabled through memory mapped I/O configuration registers
- Jump to interrupt handler from correct slot in interrupt vector table

UART interrupts

24.12.3. USART Control and Status Register 0 B

Name: UCSR0B
Offset: 0xC1
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RXCIE0: RX Complete Interrupt Enable 0
Writing this bit to one enables interrupt on the RXC0 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC0 bit in UCSR0A is set.

Bit 6 – TXCIE0: TX Complete Interrupt Enable 0
Writing this bit to one enables interrupt on the TXC0 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC0 bit in UCSR0A is set.

Bit 5 – UDRIE0: USART Data Register Empty Interrupt Enable 0
Writing this bit to one enables interrupt on the UDRE0 Flag. A Data Register Empty interrupt will be generated only if the UDRIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE0 bit in UCSR0A is set.

Interrupt vector table

16.1. Interrupt Vectors in ATmega328/P

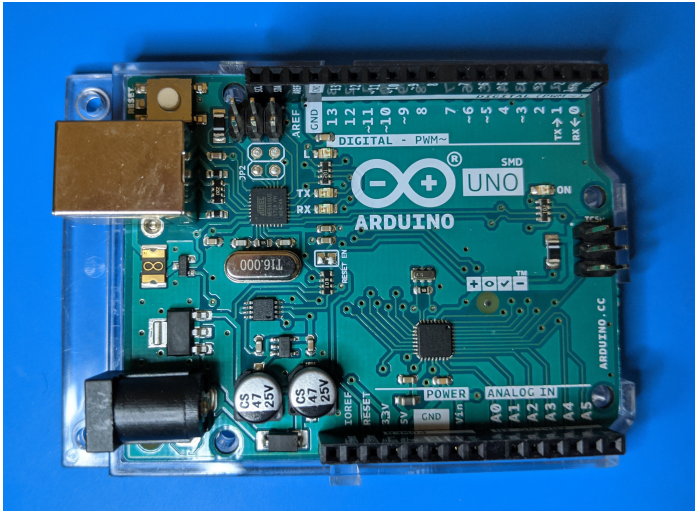
Table 16-1. Reset and Interrupt Vectors in ATmega328/P

Vector No	Program Address(2)	Source	Interrupts definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 0
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2_COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2_COMPB	Timer/Coutner2 Compare Match B
10	0x0012	TIMER2_OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1_CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1_COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1_COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1_OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0_COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0_COMPB	Timer/Coutner0 Compare Match B
17	0x0020	TIMER0_OVF	Timer/Counter0 Overflow
18	0x0022	SPI STC	SPI Serial Transfer Complete
19	0x0024	USART_RX	USART Rx Complete
20	0x0026	USART_UDRE	USART Data Register Empty
21	0x0028	USART_TX	USART Tx Complete

Quirks

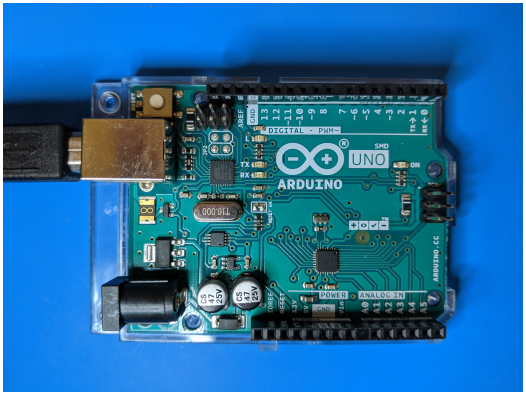
- `SP` points to first *unused* entry on stack
- No `ADDI` instruction
  - Instead, will subtract a negative value
- `R1` often used by compiler as zero register
- `MUL` destination is `R1:R0`
- How many things have an address 0?
  - Program memory address 0
  - Data memory address 0
    - Memory-mapped `R0`
  - EEPROM address 0
  - I/O register 0
    - Same as Data memory address 0x20

Arduino Uno overview



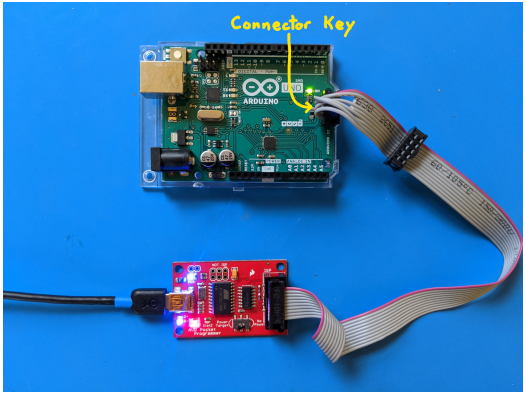
UART communications

- Connect to USB B
- Baud rate 38400



Programming

- Typical Arduino bootloader *is not* installed
- Program using ICSP header with AVR Pocket Programmer and `avrdude`



Demo

Assignment 3