Custom AVR Assembler
Caleb Jurgensen

Objectives

• Make an AVR assembler
• Program an Arduino Uno with assembly
AVR

- A family of 8-bit RISC microcontrollers developed by Atmel
  - Use the AVR architecture and instruction set
- Arduinos generally use these
  - Arduino Uno uses the atmega328p
- AVR32 is a 32 bit version

Turning on an LED

- Pins are controlled by registers
- Uno’s LED is on PB5
- Assembly

```
.include "/m328Pdef.inc"

ldi r16,0b00010000
out DDRB,r16 ; Set data direction to out
out PortB,r16  ; Set pins high or low
Start:
    rjmp Start   ; Loop
```
Making it blink

- Need a counter that counts to roughly 2 million for 1Hz blinking
- Spread the number across three registers
- add and adc set a carry flag. adc also adds this bit

Increment:
- add numLow, one ; increment lower by one
- adc numMid, zero ; add zero + carry bit to mid
- adc numUpper, zero ; same for upper
- cpi numUpper, 0b00011111 ; half of full register
- breq TurnOn ; branch if numUpper is Same or Higher than value in previous instruction
  rjmp Increment ; else increment again

```
.include "./m328Pdef.inc"
.undef ledMask = 0b00100000
.undef zero = r1
.undef numLow = r16
.undef numMid = r17
.undef numUpper = r18
.undef one = r19
.undef temp = r20

ldi one, 1 ; initialize our "one" register
ldi zero, 0 ; make sure zero register is 0
ldi temp, 0b00011111 ; half of full register
breq TurnOn ; branch if numUpper is Same or Higher than value in previous instruction
rjmp Increment ; else increment again

TurnOn:
ldi temp, ledMask
out PortB, temp ; Set led pin data direction to out
out PortB, temp ; Set led pin to high

Increment:
add numLow, one ; increment lower by one
adc numMid, zero ; add zero + carry bit to mid
adc numUpper, zero ; same for upper
breq numUpper, 0b00011111 ; half of full register
breq TurnOff ; branch if numUpper is Same or Higher than value in previous instruction
rjmp Increment ; else increment again

TurnOff:
clr numLow
clr numMid
clr numUpper
```

rjmp Increment ; start incrementing again
Arduino compiles this program
Into 385 assembly instructions

#include <Arduino.h>

int led = 13;
void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}

Assembler

• AVR Assembler
  • Doesn’t support multi-line comments

• So I made an assembler
  • Supports multi-line comments
  • Not much else
Preprocess

- Strip out comments
- Look for labels
- Look for directives
- Store stuff in arrays

```python
line = re.sub(r".*|\*/.*|\*/.*\*/", "", line)

.include "/m328Pdef.inc"

ldi r16, 0b00010000
out DDRB,r16 ; Set data direction to out
out PortB,r16 ; Set pins high or low

Start:
rjmp Start ; Loop
```

```plaintext
program[] =

<table>
<thead>
<tr>
<th>instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldi r16, 0b00010000</td>
</tr>
<tr>
<td>out DDRB, r16</td>
</tr>
<tr>
<td>out PortB, r16</td>
</tr>
<tr>
<td>rjmp Start</td>
</tr>
</tbody>
</table>

labels[] =

<table>
<thead>
<tr>
<th>Key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddrb</td>
<td>0x04</td>
</tr>
<tr>
<td>portb</td>
<td>0x05</td>
</tr>
<tr>
<td>and tons of stuff from the .inc file</td>
<td></td>
</tr>
<tr>
<td>start</td>
<td>4</td>
</tr>
</tbody>
</table>
```

Get instruction format

- Load immediate: `ldi Rd, K`
- Clear register: `clr Rd`

```
"ldi": {
  "format": "register-immediate",
  "opcode": "0b1110"
}

"register-immediate": {
  "regex": "\s*(?P<Rd>\w+)\s*(?P<K>\w+)'\s*"",
  "fields": {
    "opcode": {
      "bits": [15,14,13,12]
    },
    "Rd": {
      "bits": [7,6,5]
    },
    "K": {
      "bits": [11,10,9,8,3,2,1,0]
    }
  }
}
```

```

1110 KKKK dddd KKKK  
0010 01dd dddd dddd  
0010 01dd dddd dddd  
0010 01dd dddd dddd  
```
Evaluate fields

- Store these values in an object
- Evaluate
  - Convert to integer
  - Evaluate general purpose register
  - Look up label

```javascript
ldi r16,0b00100000
fieldValues = {
  opcode: "0b1110"
  Rd: "r16"
  K: "0b00100000"
}

fieldValues = {
  opcode: 14
  Rd: 16
  K: 32
}
```

Convert to bytecode

- Convert program to intel hex format

```plaintext
ldi r16,0b00100000  converts to    0x00E2
```

- Program data
- Data length (8 bytes)
- Address start (don’t care)
- Record type (00 = data)
- Checksum (sum instructions, take twos complement, take last byte)
- EOF
Upload with AVRDUDE

• AVR Downloader/Uploader
• Command for Arduino Uno:
  avrdude -p m328p -c arduino -P /dev/ttyUSB0 -b 115200 -U flash:w:ledOn.hex

Demo
References

- Check it out on GitHub: https://github.com/CalebJ2/avr-assembler
- https://www.microchip.com/webdoc/avrassembler/
- http://www.nongnu.org/avrdude/
- https://github.com/lpodkalicki/blog/blob/master/avr/asm/include/m328Pdef.inc
- https://regex101.com
- https://pythonhosted.org/bitstring/
- https://www.codeproject.com/Articles/712610/AVR-Assembler
- http://nuft.github.io/avr/2015/08/02/avr-hex-programming.html

Questions?