ECE 271 – Microcomputer Architecture and Applications Lecture 10

Vince Weaver

http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

21 February 2019

Announcements

- Read Chapter 2, Chapter 16

Lab#4 Notes

- Remember to disconnect your keypad, especially if you are watching the ODR lines and they aren't changing.
- You can have more than one branch jump to the same label. Labels are just placeholders for memory addresses.
- On Keil, spacing does matter for the assembly if your code starts to far to the left it will give you an error as it will think the opcode is a label

Lab#5 Preview

- Stepper motors
- Unlike regular motors, can “step” a little bit at a time and accurately set position
- To do this, we will use 4 GPIOs to control things
- The BSRR register makes it a bit easier to set/clear the GPIO pins at the same time.
- We will use 4 pins in the GPIOB register
- There will be a pattern we send on the pins that will cycle through and advance the stepper

```
         0    1    2    3    4

         :____:____:    :    :
PB2      |    :    |____:____:

         :    :    :____:____:
PB3      :____:____|

         :    :____:____:    :
PB6      :____|    :    |____:

         :____:    :    :____:
```

```
PB7     :    |____:____:    :
```

- Stepper motors used when need exact control
  Example: Disk][drive in original Apple II
  Unusual in that it was purely software controlled, leading to lots of interesting copy protection methods

Program Counter Review

```
8000010     4990  ldr r1,[pc,#256]
8000012     6ccb  ldr r3,[r1,#76]
8000014     f043
8000016     0302  orr r3,r3,#2
8000018     69c4  str r3,[r1,#76]
...
8000110     40021000 (constant)
```

PC is at 80000010, so loads the ldr instruction t

```
loads the memory value located at address of pc+2
          instruction done, incrememnts PC to 80000
PC is at 80000012, so loads the ldr instruction f
loads the memory value located at address of r1+7
instruction done, increments (this insn was 2 byt
PC is at 800000106, so loads the orr instruction
orrs the value in r3 with constant #3, stores in
instruction done, increments PC to 8000018
```

Number Representation

- Why use Base-2 in computers/digital logic?
  Why not Base-3 or Base-4? Or Base-10?
- Babbage's difference/analytical engine base-10 computer?
- Octal (useful if multiple of 3 bits), Hexadecimal (useful if multiple of 4 bits)
- Why are bytes (technically octets) 8-bits?
- What do you call 4-bits? (sometimes a nibble or nybble, a half-byte)

Unsigned Integers

- What's the biggest number you can represent?
  $2^N - 1$ so roughly 4 billion on 32-bit machine
- What happens if you overflow?
  Wraps to zero
- What *should* happen if you overflow?
  Is this an error? Should it be?
- What does C do if you overflow?
  Wraps to 0.
- What's the maximum size of adding two N bit unsigned

integers?

N+1 bits.

Signed Integers

- Sign-magnitude
  High bit is a sign bit
  Two zeros? How does that complicate things? Checking if equal?
- One's complement
  negative number is bitwise-inverse
  have to do "end-around carry" (add carry bit to rightmost bit)
- Two's complement

negative number is inverse, plus one
Can you have 9’s complement?

- What does C use?
  Implementation dependent (whatever the hardware uses)
- What does the hardware use?
  Most hardware these days is 2’s complement

Binary	Sign	One's	Two's
0000	+0	+0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

Two's complement

- Hardware for addition and subtraction is the same
  No need for special subtractor
- Addition/Subtraction/Multiplication of unsigned vs signed is mostly the same
- Is this only in binary? Can you do 9's complement with decimal?

The Carry Flag

- Unsigned addition: when two unsigned integers added, carry happens when result is too big to fit in maximum integer size $(2^n - 1)$
- Unsigned subtraction: when two unsigned integers subtracted, borrow happens when result is less than 0 (ARM has no dedciated borrow flag, carry flag is re-used)

The Overflow Flag

- Signed addition: when adding two positive numbers and wraps to being negative
- Signed addition: when adding two negative numbers and wraps to being positive
- Signed subtraction: sub pos from neg creates pos result
- sub neg from pos gettig neg result

Calculating the Overflow Flag

- Overflow occurs when the carry into the sign bit *differs* from the carry out of the sign bit

```
  5      0101
 +2      0010
=====   ====== Cout=0,C=0
  7      0111  Cin=0, V=0

  5      0101
```

```
+ 6      0110
====   ========== Cout=0,C=0
 11 (-5) 1011     Cin=1, V=1

  9    -7       1001
+10    -6      +1010
====   ===    ====== Cout=1, C=1
 19(3) -13   (1)0011 Cin=0,  V=1

 15     -1       1111
+14     -2      +1110
```

```
====    ===     ====== cout=1, C=1
 29     -3    (1)1101  Cin=1,  V=0
```

- How does the CPU know if you are doing signed vs unsigned addition?

  It doesn't. It just always sets the C and V bits.

  With two's complement it's up to you to track things if you care.
- Does the C language track the C and V bits?

Character Encodings

- ASCII – American Standard Code for Inforation Interchange

  Handy that numbers are consecutive, then lower case is offset from uppercase

  Technically 7-bit. What do you do with 8-bit? Parity? Extended characters?
- EBCDIC?
- Unicode? 16-bit?
- UTF-8?

- Emojis?