Today's Topics

Odds and ends I haven't had a time to talk about.

- Strings and bytes.
- syscalls
- Some example programs.

Strings

- On most machines characters are stored as single bytes.
- Each character is given a unique number.
- For example ’A’ has the code 0x41.
- The most popular coding is ASCII code and its various extensions.

Strings

A string is just a sequence of characters.

- How do you know how long a string is?

The are at least two solutions to the problem:

- Store the length of the string at the beginning.
- Put a terminating character at the end.

Strings

- C, the MIPS syscall library and many languages chose to store a terminating character at the end of the string.
- The terminating character is code 0 or `'\0'`.

`lb` and `sb`

When manipulating strings (and other things) you need to address bytes.

- `lb $reg,offset($reg)`
- `sb $reg,offset($reg)`

Example the length of a string

```
        .data
str:    .asciiz " Hello world." ;
        .text
        .globl main

main:   addi $s0,$zero,0
        la $t0,str # $to points to current place in the string.
loop:   lb $t1,0($t0)
        beq $t1,$zero,exit
        addi $s0,$s0,1
        addi $t0,$t0,1
        j loop
exit:   jr $31
```

System calls (sycalls)

SPIM provides a small set of operating system like services through the `syscall` instruction.

To request a service a program load the system call code into register `$v0` and arguments in `$a0-$a3`. System calls that return values put the result in register `$v0`.

Example

```
   .data
str: .asciiz "the answer = "
   .text
   li $v0,4 #system call code for print_str
   la $a0,str
   syscall
   li $v0,1 #print_int.
   li $a0,5
   syscall
```

Note you are not supposed to use `print_int` in your assignment.

The best reference is Appendix A of the old Patterson & Henessy book, avaliable online from the SPIM authors page.

Setting Bits in words

Suppose we have a bit pattern `01010000` = `0x50` how do we set the bottom bit to 1?

Remember the truth table for `or`

$$\begin{aligned} 0 \;\texttt{or}\; 0 &= 0 \\ 1 \;\texttt{or}\; 0 &= 1 \\ 0 \;\texttt{or}\; 1 &= 1 \\ 1 \;\texttt{or}\; 1 &= 1 \end{aligned}$$

Setting Bits in words

Thus to set the first bit of a bit pattern `01010000` = `0x50` we simply use `or` :

```
01010000 or 00000001
```

is equal to

```
01010001
```

Setting Bits in words

Remember the `or` instruction works bitwise on numbers. So to set the third but of a binary number `x` we simple set `x` to

$$\texttt{x or } 0x04$$

You can use this method in a high-level language as well.

Setting Bits in words

Testing bits

How do we test the if bit 3 is set in a particular number?

Remember the truth table for `and`

$$\begin{aligned}
0 \ \texttt{and} \ 0 &= 0 \\
1 \ \texttt{and} \ 0 &= 0 \\
0 \ \texttt{and} \ 1 &= 0 \\
1 \ \texttt{and} \ 1 &= 1
\end{aligned}$$

Setting Bits in words

Thus if we and number with `0x04`

```
01010000 and 00000100
```

we will get zero if bit 3 is set and `0x04` otherwise.

```
01011100 and 00000100
```

Examples on the boards might possibly include :-

- fib(n)=fib(n-1)+fib(n-2), fib(0)=0,fib(1)=1.
- Various things to do with strings.