RICHARD ROBINSON

ASSEMBLY LANGUAGE

Contents

Introduction

The Basics

In programming, a compiler is a program that translates high-level language statements into assembly language statements. In turn, an assembler translates assembly language into machine language in binary for MIPS. The MIPS unit is given by

$$\text{MIPS} = \frac{\text{clockrate}}{\text{CPI} \times 10^6} \tag{1}$$

Data Transfer Instruction: a command that moves data between memory and registers.

In MIPS, arithmetic operations may only be single and has three variables. For example,

```
add $s1, $s2, $s3
```

Address: a value used to delineate the location of a specific data element within a memory array

is equivalent to `$s1 = $s2 + $s3` where `$si` are registers for variables. To access a word in memory, the instruction must supply the memory address. Since memory is an array, the address is an index to such.Since MIPS addresses each byte, word addresses are multiples of 4 as there are 4 bytes per word. That is, `lw $t0,4n` gets `A[n]`.

Assume `h` is associated with reg `$s2` with base address of `A` is in `s3`. For the C statement

```
A[x] = h + A[y]
```

the equivalent MIPS code is

```
lw   $t0,4y($s3) # Temp reg $t0 gets A[8]
add  $t0,$s2,$t0 # Temp reg $t0 gets h + A[8]
sw   $t0,4x($s3) # Stores h + A[y] into A[x]
```

Binary & Hexadecimal

The conversion from a binary number `n...n` where `x` is `0,1` and x_i is the ith bit of x is

$$\sum x_i 2^i \tag{2}$$

from right to left, with the rightmost bit being the least significant bit. Therefore, there are $2^{32} - 1$ binary numbers in 32 bits unsigned.

The **two's complement** representation has the binary complements related by

$$\overline{x} + 1 = -x \tag{3}$$

In the instruction format, each instruction comprises 32 bits into six segments, with the first and last segments having six bits and the others having five which instructs MIPS. These fields are

```
op   rs   rt   rd   shamt   funct
```

These fields perform the following functions:

- `op`: Basic op of the instruction
- `rs`: The first reg src op
- `rt`: The second reg src op
- `rd`: The reg dest op
- `shamt`: Shift amount
- `funct`: Selects variant of `op`

Hexadecimal is an extension of decimal which extends to 16 digits (1-9,a-f). Thus, any permutation of four bits may be represented by a single hexadecimal digit.

Logical Operators

Shifting left by i bits gives the same result as multiplying by $2i$. The `and` and `or` operators operate as their logical counterparts. The `nor` operator is such that `A NOR 0 = NOT (A OR 0) = NOT (A)`. Specifically,

```
nor $t0,$t1,$t3 # reg $t0 = ~ (reg $t1 | reg $t3)
```

For constants, the operators `andi` and `ori` are used. For example,

```
if (i == j) f = g + h; else f = g - h
```

in MIPS is translated to

```
bne $s3,$s4,Else       # go to Else if i != j
add $s0,$s1,$s2        # if i == j then f = g + h
j Exit                 # jump to Exit
Else:sub $s0,$s1,$s2   # f = g - h else
Exit:
```

`<<`	`sll`
`>>>`	`srl`
`&`	`and`
`|`	`or`
`~`	`nor`

Java and MIPS equivalent logical operations.