Next: Link to Subroutines **Up:** Chapter 4: Instruction Set **Previous:** Instruction Set of MIPS

Many Examples:

- Assume you are the compiler which translates the following C program segments into MIPS assembly programs composed of instructions. (Any thing following ``#'' is the comment, not part of the program.)
- It is the compiler's job to assign a distinct register to each variable used in the C program. Here we assume registers $s0 (16), $s1 ($17), $s2 ($18), $s3 ($19), $s4 ($20), $s5 ($21) are assigned to contain variables f, g, h, i, j, k, respectively:

$s0 ($16)	$s1 ($17)	$s2 ($18)	$s3 ($19)	$s4 ($20)	$s5 ($21)
f	g	h	i	j	k

- It is also the compiler's job to link labels and symbolic addresses used in the assembly program to actual memory addresses (to be discussed later).

1. The C program:

   f = (g+h)*(i+j)

   The MIPS assembly:

```
add $t0, $s1, $s2   # a temporary register $8 contains g + h
add $t1, $s3, $s4   # a temporary register $9 contains i + j
mul $s0, $t0, $t1   # f gets (g + h) * (i + j)
```

2. The C program:

   A[i]=h+A[i]

   The MIPS assembly:

```
lw $t0, Astart($s3)   # load A[i] into temporary register $t0
add $t0, $s2, $t0     # temporary register $8 gets g=h+A[i]  l
sw $t0, Astart($s3)   # store h+A[i] into A[i]
```

   Note: (0) The starting address of array A is assumed to be Astart. (1) Remember $s3 (19) contains variable i, (2) If the elements of the array are words instead of bytes, i should be multiplied by 4 to point to the proper word address in the memory.

3. The C program:

$$for\ (i = 0;\ \ i < k;\ \ i++) \quad h = h + A[i]$$

The MIPS assembly:

```
        add $s3, $zero, $zero   # set i=0
loop    lw $t0, Astart($s3)     # load A[i] into $t0
        add $s2, $s2, $t0       # h=h+A[i]
        addi $s3, $s3, 1        # i=i+1
        bne $s3, $s5, loop      # loop if i < k
        ... ... ... ...
```

4. The C program:

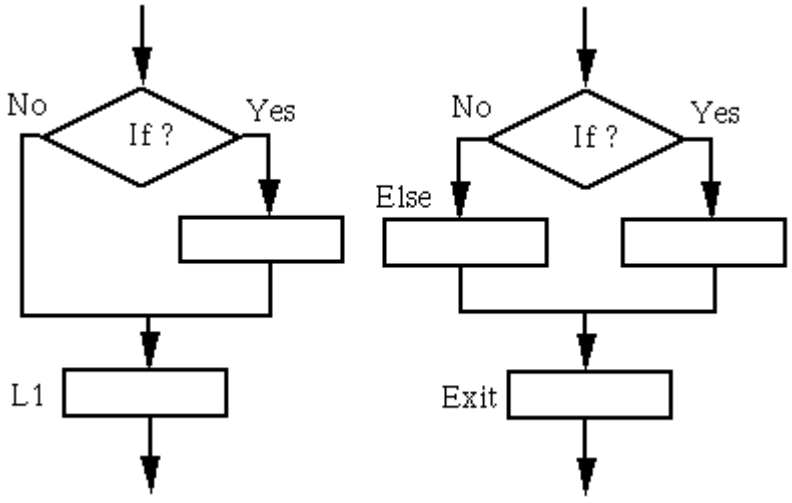
```
if ( i!=j ) f = g - h;
f = f + i;
```

or, equivalently,

```
     if ( i==j ) goto L1;
     f = g - h;
L1:  f = f + i;
```

The MIPS assembly:

```
        beq $s3, $s4, L1    # goto L1 if i = j
        sub $s0, $s1, $s2   # f = g - h if i ≠ j
L1:     add $s0, $s0, $s3   # f = f + i is always executed
```



5. The C program:

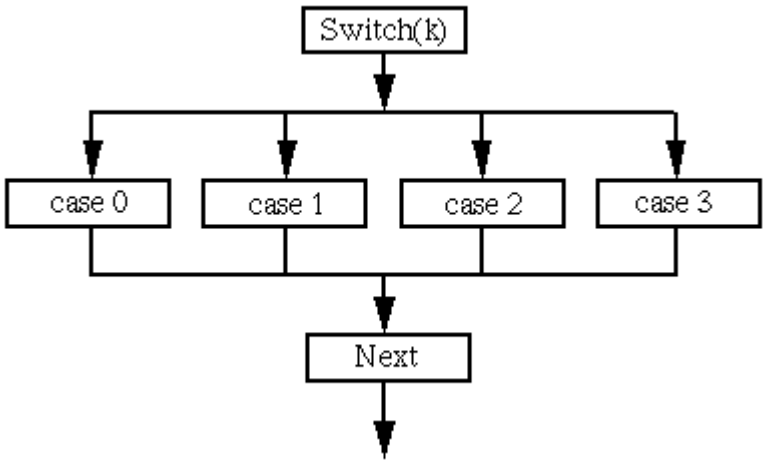
```
if ( i==j ) f=g+h;
else f=g-h;
```

The MIPS assembly:

```
      bne $s3, $s4, Else   # goto Else if i is not equal j
      add $s0, $s1, $s2    # f=g+h if i equals j
      j Exit               # goto Exit
Else: sub $s0, $s1, $s2    # f=g-h if i is not equal to j
Exit: ...                  # next instruction
```

6. The C program:

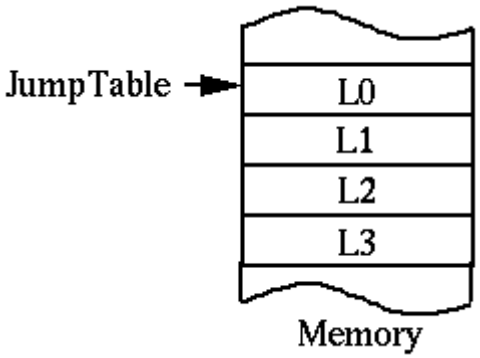
```
switch (k) {
case 0: f=i+j; break
case 1: f=g+h; break
case 2: f=g-h; break
case 3: f=i-j; break
}
```



Assume labels L0 through L3 are stored in 4 memory locations starting at JumpTalbe.

The MIPS assembly:

```
        lw $t0, JumpTable($s5)   # temp reg $t0 gets JumpTable[k]
        jr $t0                   # jump to where $8 points to
L0:     add $s0, $s3, $s4        # k=0, so f=i+j
        j Exit                   # end of case
L1:     add $s0, $s1, $s2        # k=1, so f=g+h
        j Exit                   # end of case
L2:     sub $s0, $s1, $s2        # k=2, so f=g-h
        j Exit                   # end of case
L3:     sub $s0, $s3, $s4        # k=3, so f=i-j
        j Exit                   # end of case
Exit:   ...                      # next instruction
```



Next: Link to Subroutines **Up:** Chapter 4: Instruction Set **Previous:** Instruction Set of MIPS
Ruye Wang 2003-10-30