MIPS assembly

Review

- We learned
  - `addi`,
  - `and`, `andi`, `or`, `ori`, `xor`, `xori`, `nor`,
- An array is stored sequentially in the memory
- The instructions are also stored sequentially in the memory. Executing the code is to load then execute the instructions one by one, unless we encounter a branch condition.

Shifts

- Shift instructions move all the bits in a word to the left or to the right
  - Shift left logical (sll) move all the bits to the left by the specified number of bits
    - sll $t2, $t0, 2
  - Shift right logical (srl) move all the bits to the right
    - srl $t2, $t0, 2
  - Filling the emptied bits with 0's
    - This includes srl with negative numbers (since you insert 0's to the left of the number, your number will be positive after the shift)

Example 1

- Suppose register \$s0 (\$16) is 9_{ten}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

  – What do we have in \$t2 (\$10) after `sll $t2, $s0, 4`

Example 1

- Suppose register $s0 ($16) is 9_{ten}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

  – We have in $t2 ($10) after `sll $t2, $s0, 4`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

    - The value is $144_{ten} = 9_{ten} \times 2^4$

  – In general, shifting left by i bits gives the same result as multiplying by 2^i

Example 2

- Suppose register \$s0 (\$16) is 9_{ten}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

  – What do we have in \$t2 (\$10) after `sll $t2, $s0, 28`

Example 2

- Suppose register $s0 ($16) is 9_{ten}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

  – We have in $t2 ($10) after `sll $t2, $s0, 28`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

    - The value is NOT $9_{ten} \times 2^{28}$ noting that the number is a signed number.
    - Overflow happens this time

Example 3

- Suppose register \$s0 (\$16) is 99_{ten}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1

  – What do we have in \$t2 (\$10) after srl \$t2, \$s0, 4

Example 3

- Suppose register \$s0 (\$16) is 99_{ten}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1

  – We have in \$t2 (\$10) after srl \$t2, \$s0, 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

    - The value is $6_{ten} = 99_{ten} / 2^4$

  – In general, shifting left by i bits gives the same result as dividing by 2^i

Example 4

- Suppose register \$s0 (\$16) is -9_{ten}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1

  – What do we have in \$t2 (\$10) after srl \$t2, \$s0, 4

Example 4

- Suppose register $s0 ($16) is -9_{ten}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1

  – We have in $t2 ($10) after srl $t2, $s0, 4

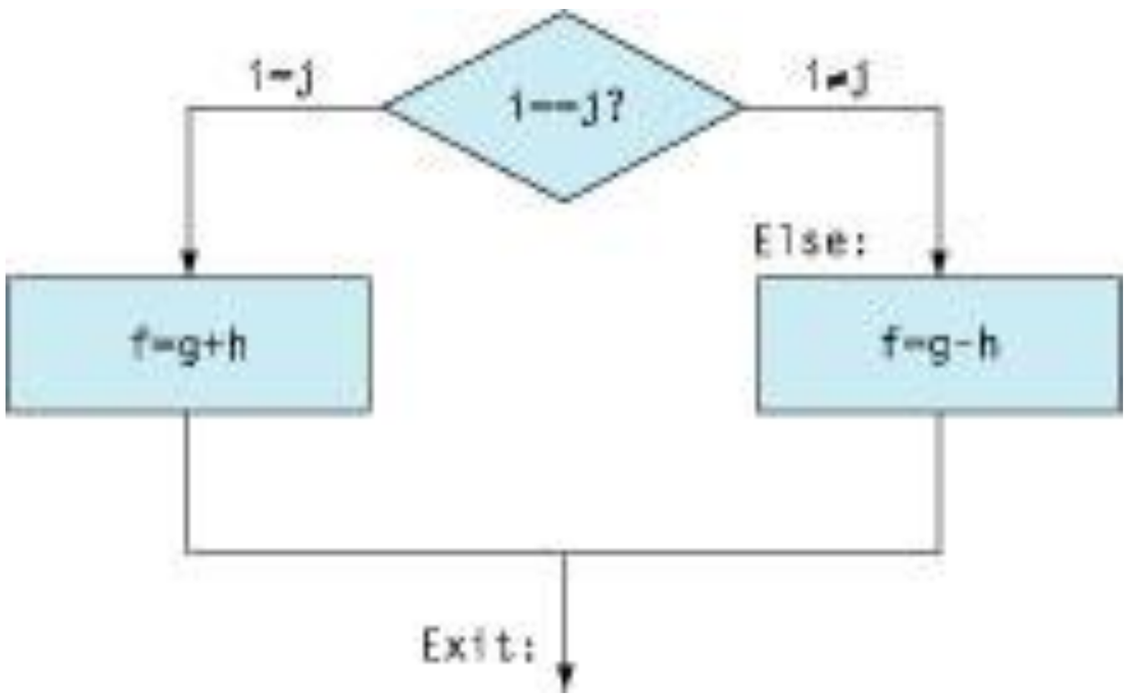
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

    - The value is NOT $-9_{ten} / 2^4$ noting that the number is a signed number.
    - Even though it's a negative number, 0's are filled in during shift

Instructions for Making Decisions

- A distinctive feature of programs is that they can make different decisions based on the input data

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



Instruction beq (branch if equal)

- To support decision making, MIPS has two conditional branch instructions, similar to an "if" statement with a goto

```
beq register1, register2, L1
```

  - In C, it is equivalent to

```
if (register1 == register2)
  goto  L1
```

  - Note that L1 is a label and we are comparing values in register1 and register2
- Label is **an address of an instruction**.
  - Every address can be associated with a label, which is used by the assembly program to specify the address
  - Go to a label means that fetch that instruction from the memory and execute it.

Instruction bne

- Similarly, bne (branch not equal) means go to the statement labeled with L1 if the value in register1 does not equal to the value in regster2

```
bne register1, register2, L1
```

  – Equivalent to

```
if (register1 != register2)
  goto  L1
```

Instruction j (jump)

- MIPS has also an unconditional branch, equivalent to goto in C

```
j   L1
```

  – Jump to the instruction labeled with L1

Compiling if-then-else

- Suppose variables f, g, h, i, and j are in registers $s0 through $s4, how to implement the following in MIPS?

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

Compiling if-then-else

- Suppose variables f, g, h, i, and j are in registers $s0 through $s4, how to implement the following in MIPS?

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

```
      if (i != j)
         goto Else;
      f = g + h;
      goto Exit;
Else:
      f = g - h;
Exit:
```

Compiling if-then-else

- Suppose variables f, g, h, i, and j are in registers $s0 through $s4, how to implement the following in MIPS?

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

```
      if ($s3 != $s4)
         goto Else;
      $s0 = $s1 + $s2;
      goto Exit;
Else:
      $s0 = $s1 - $s2;
Exit:
```

MIPS Assembly for if-then-else

- Now it is straightforward to translate the C program into MIPS assembly

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

```
     bne $s3,$s4,Else;       #go to Else if i <> j
     add $s0, $s1, $s2       #f = g + h
     j   Exit;               #go to the end of the if-then-else block
Else:
     sub $s0, $s1, $s2       #f = g -h
Exit:
```

Exercise 1

- Suppose $t0 is storing 30, $t1 is storing 20. After the following instructions, what will be the value in $t2?

```
sub $t2, $t0, $t1
srl $t2, $t2, 2
ori $t2, $t2, 10
```

(a) 8

(b)10

(c)18

(d) None of the above.

Exercise 2

- Suppose word array A stores 0,1,2,3,4,5,6,7,8,9, in this order. Assume the starting address of A is in $s0. After the following instructions, what will be the value in $t0?

```
addi $s0, $s0, 32
lw $t0, 4($s0)
andi $t0, $t0, 1
```

(a) 0
(b) 8
(c) 9
(d) None of the above.

Exercise 3

- If $t0 is holding 17, $t1 is holding 8, what will be the value stored in $t2 after the following instructions?

```
      andi $t0, $t0, 3
      beq $t0, $0, L1
      addi $t0, $t0, 1
L1:   add $t2, $t0, $t1
```

(a) 10.

(b) 8.

(c) 2.

(d) None of the above.

Exercise 4

- Assume A is an integer array with 10 elements storing 0,1,2,3,4,5,6,7,8,9. Assume the starting address of A is in $s0 and $t0 is holding 3. After the running the following code, what will be the content of $t0?

```
sll $t0, $t0, 3
add $t0, $s0, $t0
lw $t0, 0($t0)
srl $t0, $t0, 1
```

(a) 3
(b) 1
(c) 0
(d) None of the above.

In Class Exercise

- If-Else