CONTROL INSTRUCTIONS

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

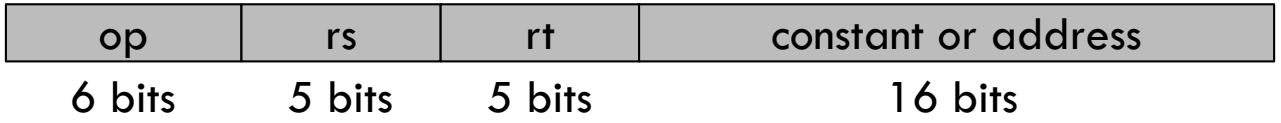
Overview

- This lecture
  - Control Instructions
  - Programming guidelines
    - If-else
    - Do-while
    - For-loop

MIPS Instruction Format

- Instructions are represented as 32-bit numbers with multiple fields
- MIPS Instruction Types
  - R-type

  - I-type



  - J-type

Control Instructions

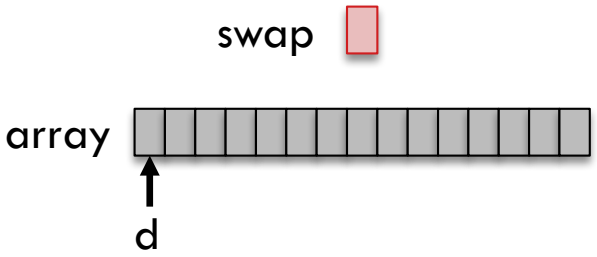
- We need decision making instructions to control the execution flow
  - Example C code

```
4  for (c = 0 ; c < n - 1; c++) {
5      for (d = 0 ; d < n - c - 1; d++) {
6          if (array[d] > array[d+1]) {
7              swap       = array[d];
8              array[d]   = array[d+1];
9              array[d+1] = swap;
10         }
11     }
12 }
```

Control Instructions

- We need decision making instructions to control the execution flow
  - Example C code

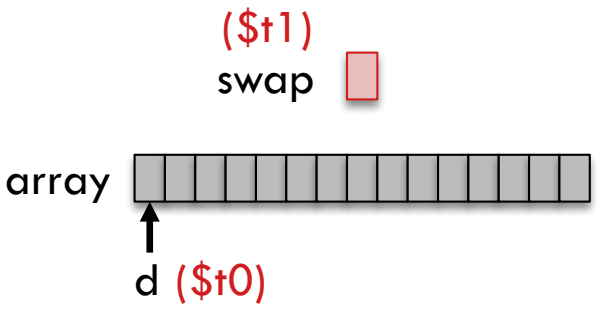
```
4  for (c = 0 ; c < n - 1; c++) {
5      for (d = 0 ; d < n - c - 1; d++) {
6          if (array[d] > array[d+1]) {
7              swap       = array[d];
8              array[d]   = array[d+1];
9              array[d+1] = swap;
10         }
11     }
12 }
```



Control Instructions

- We need decision making instructions to control the execution flow
  - Example C code

```
4  for (c = 0 ; c < n - 1; c++) {
5      for (d = 0 ; d < n - c - 1; d++) {
6          if (array[d] > array[d+1]) {
7              swap       = array[d];
8              array[d]   = array[d+1];
9              array[d+1] = swap;
10         }
11     }
12 }
```

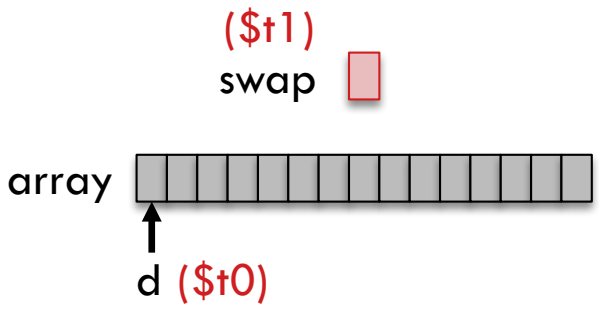


Control Instructions

- We need decision making instructions to control the execution flow
  - Example C code

```
4  for (c = 0 ; c < n - 1; c++) {
5      for (d = 0 ; d < n - c - 1; d++) {
6          if (array[d] > array[d+1]) {
7              swap       = array[d];
8              array[d]   = array[d+1];
9              array[d+1] = swap;
10         }
11     }
12 }
```

lw $t1, 0($t0)

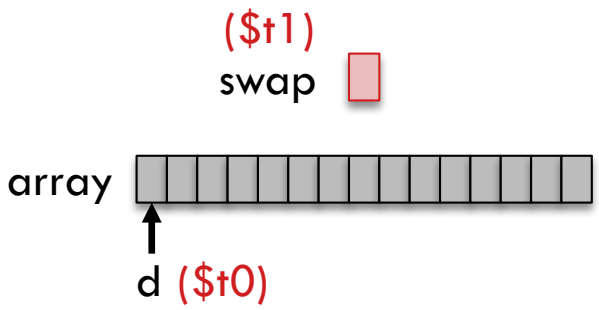


Control Instructions

- We need decision making instructions to control the execution flow
  - Example C code

```
4  for (c = 0 ; c < n - 1; c++) {
5      for (d = 0 ; d < n - c - 1; d++) {
6          if (array[d] > array[d+1]) {
7              swap       = array[d];
8              array[d]   = array[d+1];
9              array[d+1] = swap;
10         }
11     }
12 }
```

```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
sw   $t2, 0($t0)
```

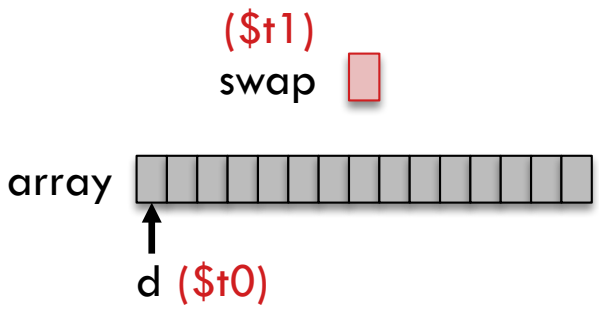


Control Instructions

- We need decision making instructions to control the execution flow
  - Example C code

```
4  for (c = 0 ; c < n - 1; c++) {
5      for (d = 0 ; d < n - c - 1; d++) {
6          if (array[d] > array[d+1]) {
7              swap       = array[d];
8              array[d]   = array[d+1];
9              array[d+1] = swap;
10         }
11     }
12 }
```

```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
sw   $t2, 0($t0)
sw   $t1, 4($t0)
```



Control Instructions

- We need decision making instructions to control the execution flow
  - Example C code

```
4  for (c = 0 ; c < n - 1; c++) {
5      for (d = 0 ; d < n - c - 1; d++) {
6          if (array[d] > array[d+1]) {
7              swap       = array[d];
8              array[d]   = array[d+1];
9              array[d+1] = swap;
10         }
11     }
12 }
```

```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
sw   $t2, 0($t0)
sw   $t1, 4($t0)
```

How to handle loops and if statements?

Control Instructions

- Determine which instruction to be executed next
  - Conditional branch: Jump to instruction L1 if register1 equals register2
    - beq register1, register2, L1
  - Unconditional branch: Jump to instruction L1
    - J L1

Control Instructions

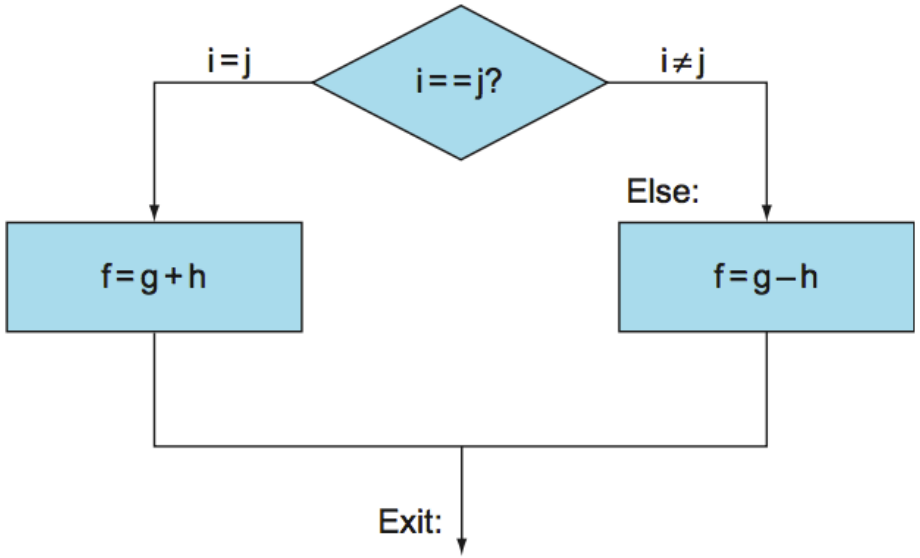
- Determine which instruction to be executed next
  - Conditional branch: Jump to instruction L1 if register1 equals register2
    - beq register1, register2, L1
    - bne, slt (set-on-less-than), slti
  - Unconditional branch: Jump to instruction L1
    - J L1
    - Jr $s0 (jump table; long jumps and case statements)

Example: If-Else

- Convert to assembly
  - if (i == j)
    - f = g + h;
  - else
    - f = g – h;

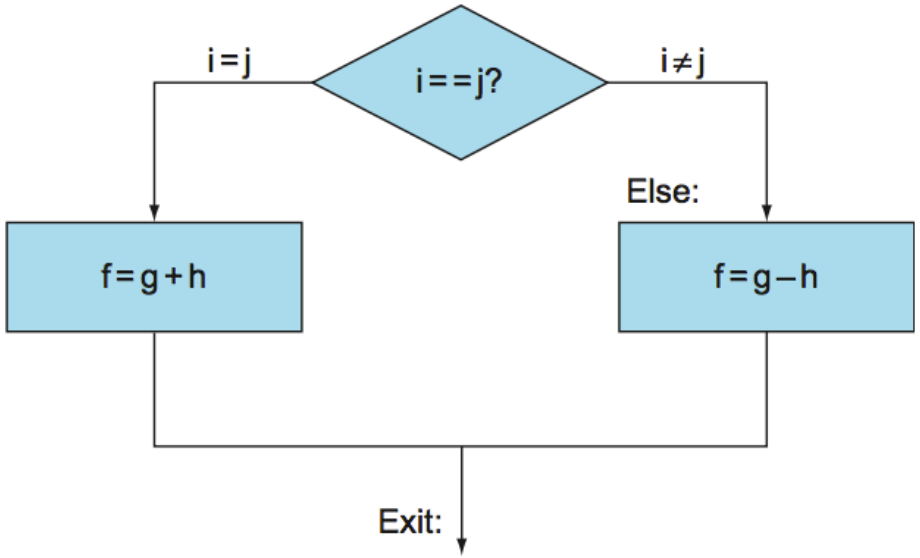
Example: If-Else

- Convert to assembly
  - if (i == j)
    - f = g + h;
  - else
    - f = g – h;



Example: If-Else

- Convert to assembly
  - if (i == j)
    - f = g + h;
  - else
    - f = g – h;



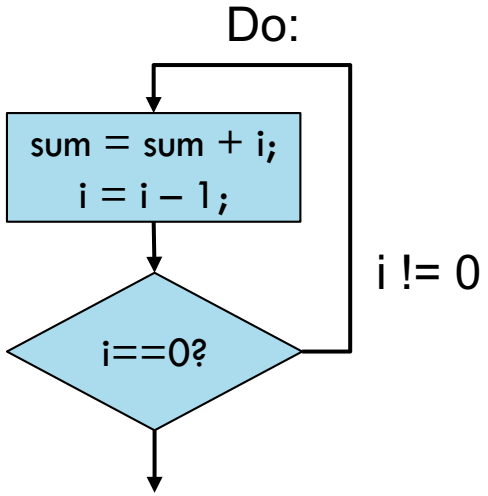

```
        bne   $s3, $s4, Else
        add   $s0, $s1, $s2
        j     Exit
Else:   sub   $s0, $s1, $s2
Exit:
```

Example: Do-While

- Convert to assembly
  - do {
    - sum = sum + i;
    - i = i – 1;
  - }while (i != 0);

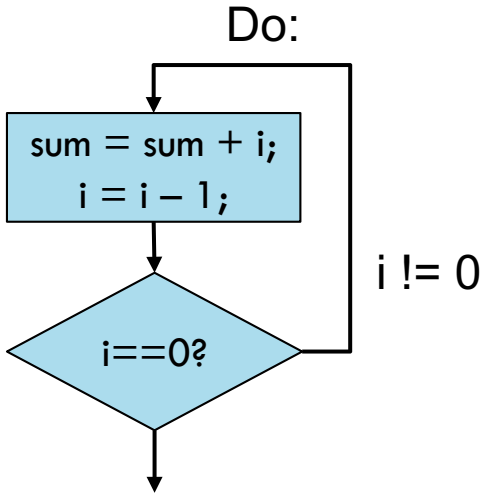
Example: Do-While

- Convert to assembly
  - do {
    - sum = sum + i;
    - i = i – 1;
  - }while (i != 0);



Example: Do-While

- Convert to assembly
  - do {
    - sum = sum + i;
    - i = i – 1;
  - }while (i != 0);



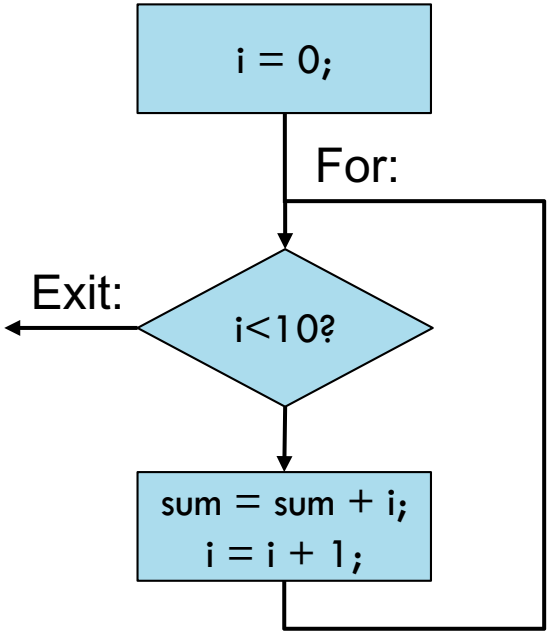

```
Do:     add   $s0, $s0, $t0
        subi  $t0, $t0, 1
        bne   $t0, $zero, Do
```

Example: For-Loop

- Convert to assembly
  - for(i=0; i<10; i=i+1) {
    - sum = sum + i;
  - }

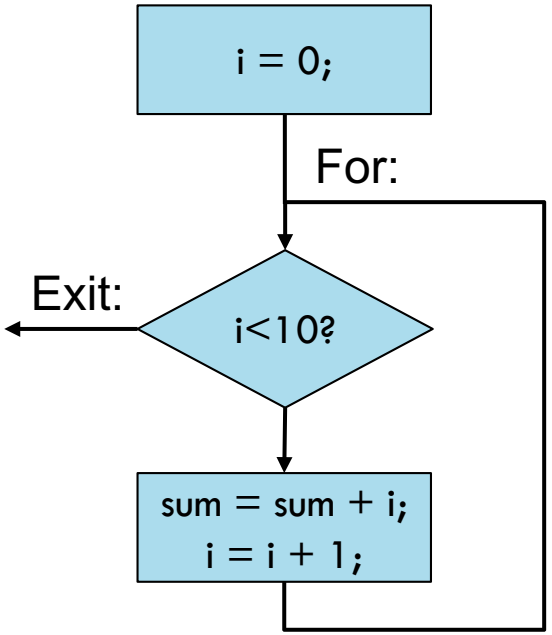
Example: For-Loop

- Convert to assembly
  - for(i=0; i<10; i=i+1) {
    - sum = sum + i;
  - }



Example: For-Loop

- Convert to assembly
  - for(i=0; i<10; i=i+1) {
    - sum = sum + i;
  - }




```
        addi  $t0, $zero, 0
For:    slti  $t1, $t0, 10
        beq   $t1, $zero, Exit
        add   $s0, $s0, $t0
        addi  $t0, $t0, 1
        j     For
Exit:
```