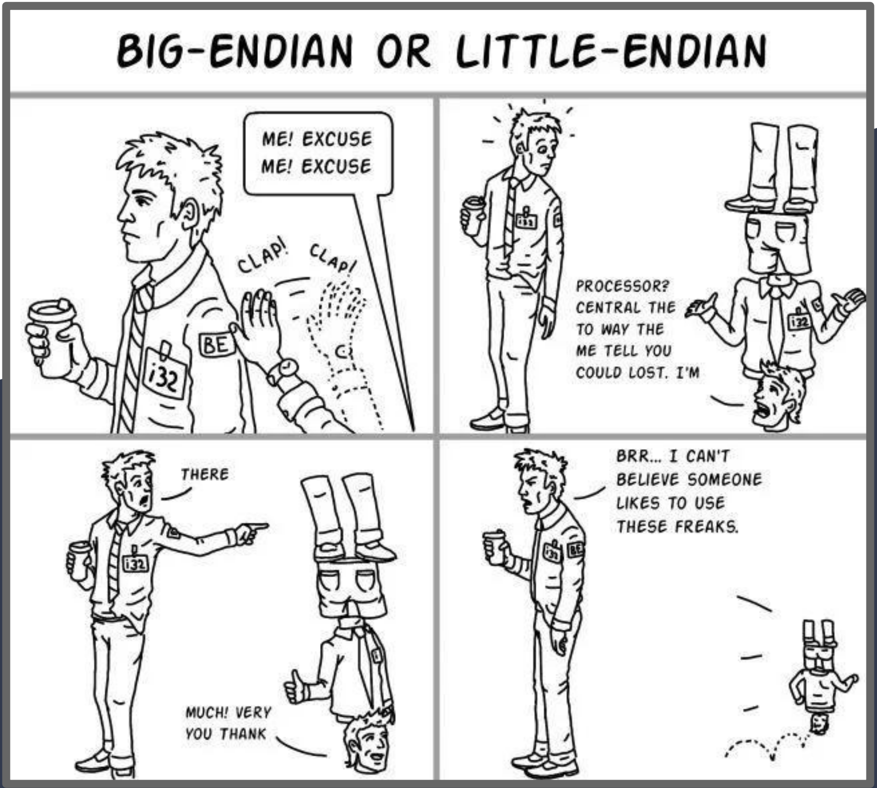
CS 2340 - Computer Architecture

6 Data Arrays, Conditional Decisions
Dr. Alice Wang

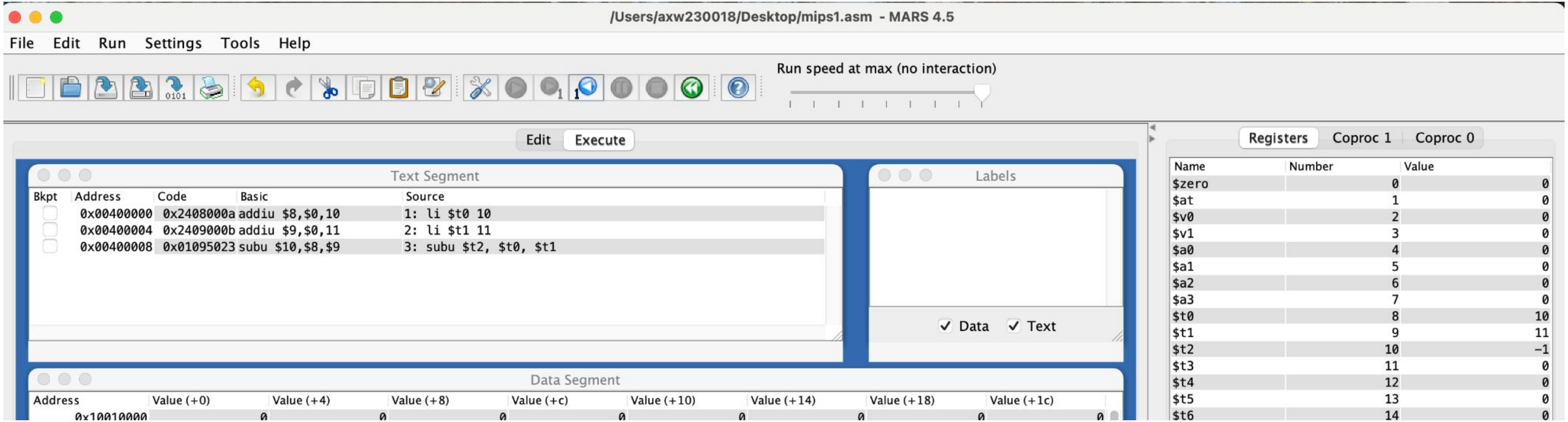


Research

`subu` - Unsigned subtraction instruction, assumes operands are unsigned.

What happens when you get a negative number after using subu?

Result is stored as two's complement signed number



Term Project released

Topic

The topic for the term project this semester is the ***Binary*** game.

For a feel and specifications of the game visit:

https://learningnetwork.cisco.com/s/binary-game

The program allows the user to solve two types of conversion problems in each round:

1. In binary-to-decimal mode, the game displays eight boxes containing either '0' or '1', along with a blank box for the user to input the decimal equivalent.
2. In decimal-to-binary mode, the game shows eight blank boxes for binary input and a decimal number in the final box.

Note: MARS does not have a graphic capability, so an ASCII characters-based board is sufficient.

Minimum requirements for the program:

- The game is for one player against the computer.
- The game must be random – each time you play the game you get different problems to solve.
- The game board is displayed using ASCII characters (e.g., -, +, and |) is the minimum requirement. Creative ways to display the board, e.g. with graphics, will earn extra credits.
- You must do binary/decimal validation and indicate to the player if an answer is invalid.
- Unlike the online game, your project does not need a timeout feature. You should implement 10 levels, with each level adding one line (e.g., Level 1 has 1 line, Level 2 has 2 lines, etc.).

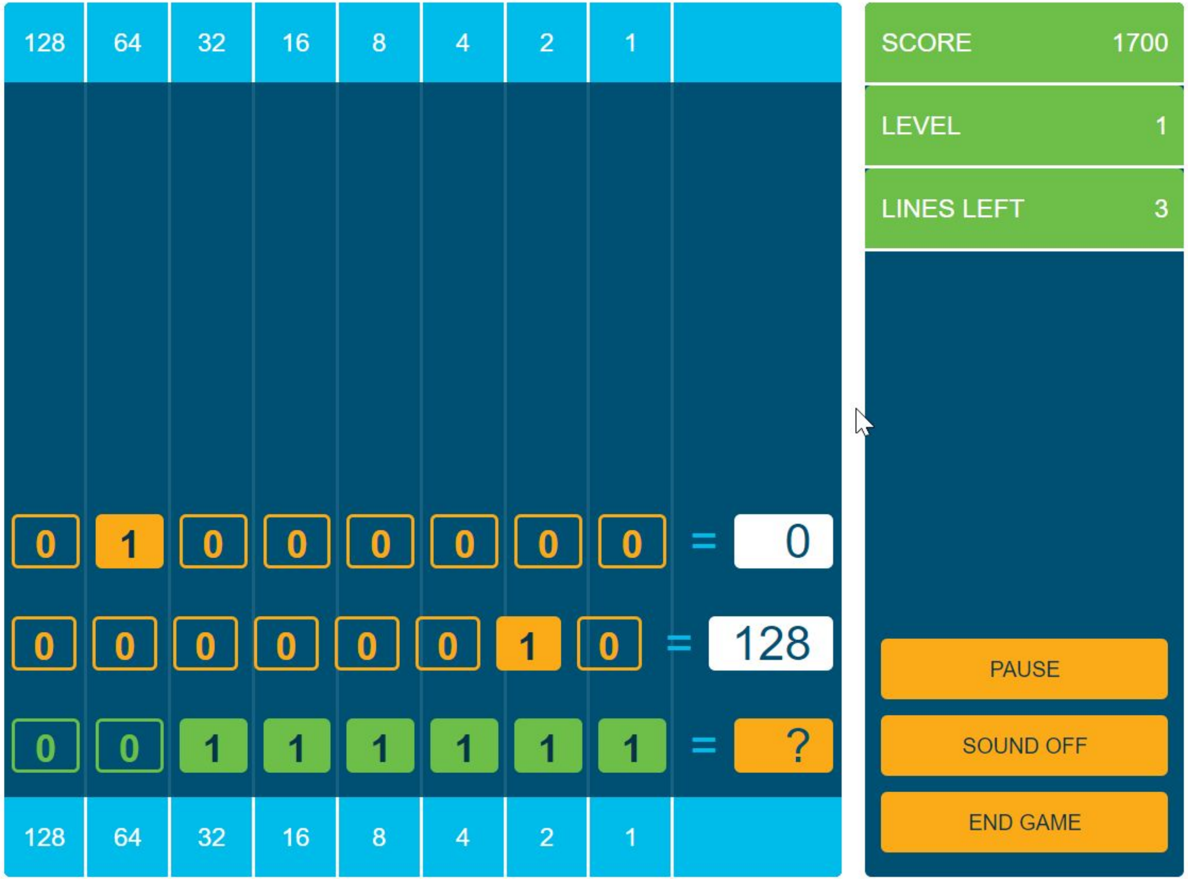
Extra credits will be given for implementing:

- Graphic (5 pts)
- Sound (5 pts)
- Timeout feature (5 pts)

https://learningnetwork.cisco.com/s/binary-game

This is an individual project (not a team project)

Term Project released – what it looks like



Term Project submission

Each student will need to submit their own assignment

1. Written report
2. Assembly code - each student should have their own unique version of the code
3. User Manual
4. Oral interview with the grader

60% implementation (verified through oral interview), 20% documentation, 20% demonstration

Due: 10/24. Do not wait until the last minute to start this project!

Review of Last Lecture

- Arithmetic Operations
  - `add`, `sub`, `addi`
  - `addu`, `subu`, `addiu`
- Memory Operations
  - `lw` (load word), `sw` (store word)
  - `lb` (load byte), `sb` (store byte)
- MIPS uses byte-addressable memories
  - Word address = multiply the array element by 4

Arrays of Words

- Allow access large amounts of similar data
  - **Index**: access each element
  - **Size**: number of elements
- Example: 5-element array of words
- **Base address** = 0x12348000 (address of first element, array[0])
- First step in accessing an array: load base address into a register (`la`)
- Next step use load word (`lw`) or store word (`sw`) to read or write from base address + offset

0x12348000	array[0]
0x12348004	array[1]
0x12348008	array[2]
0x1234800C	array[3]
0x12348010	array[4]

Array vs Pointers

- Array indexing involves
  - Multiplying index by element size
  - Adding to array base address
- Pointers correspond directly to memory addresses
  - Can avoid indexing complexity

Array vs Pointer Example

Clearing an array: Using array indexing	**Using pointers**

`# Loop updates the array index`
`clear1(int array[], int size) {`
`  int i;`
`  for (i = 0; i < size; i += 1)`
`    array[i] = 0;`
`}`
	`# Loop updates the pointer`
	`clear2(int *array, int size) {`
	`  int *p;`
	`  for (p = &array[0]; p < &array[size]; p = p + 1)`
	`    *p = 0;`
	`}`

`       la $a0, array        # base add. Array`
`       li $a1, size         # assume size in $a1`
`       move $t0,$zero       # i = 0`
`loop1: mul $t1,$t0,4        # $t1 = i * 4`
`       add $t2,$a0,$t1      # $t2 =&array[i]`
`       sw $zero, 0($t2)     # array[i] = 0`
`       addi $t0,$t0,1       # i = i + 1`
`       slt $t3,$t0,$a1      # $t3 = (i < size)`
`       bne $t3,$zero,loop1 # if $t3!=0,goto loop1`
	`       la $a0, array        # base add. Array`
	`       li $a1, size         # assume size in $a1`
	`       move $t0,$a0         # p = &array[0]`
	`       mul $t1,$a1,4        # $t1 = size * 4`
	`       add $t2,$a0,$t1      # $t2 =&array[size]`
	`loop2: sw $zero,0($t0)      # Memory[p] = 0`
	`       addi $t0,$t0,4       # p = p + 4`
	`       slt $t3,$t0,$t2      # $t3 =(p<&array[size])`
	`       bne $t3,$zero,loop2 # if $t3 != 0, goto loop2`

- Pointer version has fewer instructions in the loop → modern compilers will do the optimization for you
- By the end of this class you will be able to understand this loop code!

Strings

- Strings are another example of arrays, but at the byte- or character-level

A = “Hello World!”

A[0] = ‘H’

A[5] = ‘ ‘

A[8] = ____

String Manipulation

- Programming technique that involves changing or processing text data, aka strings
- Example of things we do to strings
  - **Concatenation**: Joining two or more strings together
  - **Substring extraction**: Extracting a sequence of characters from a larger string
  - **Case transformation**: Changing uppercase characters to lowercase, or vice versa
  - **Replacement**: Replacing a substring with another or deleting it
  - **Splitting**: Splitting a string into pieces at a specific character
  - **Slicing**: Extracting a substring by specifying start and end points

String Manipulation – MIPS instructions

- Use bytewise memory operations
  - `lb rt, offset(rs)      # load byte`
  - `sb rt, offset(rs)      # store byte`

Let’s do an example!

String Manipulation Example

- MyString = `“Hello! My name is Tim\n”`
- Using MIPS replace substring “T” with “J”
- Substring = MyString[18] = `“T” => “J”` (byte-index)

```
.data
MyString: .asciiz  "Hello! My name is Tim\n"

.text
        la $a0, MyString
        li $t1, 'J'             # Load character ‘J’ into $t1
        addi $t2, $a0, 18       # Add 18 to the base address of MyString
        sb $t1, 0($t2)          # Store ‘J’ to index 19 of MyString
```

String Manipulation Example

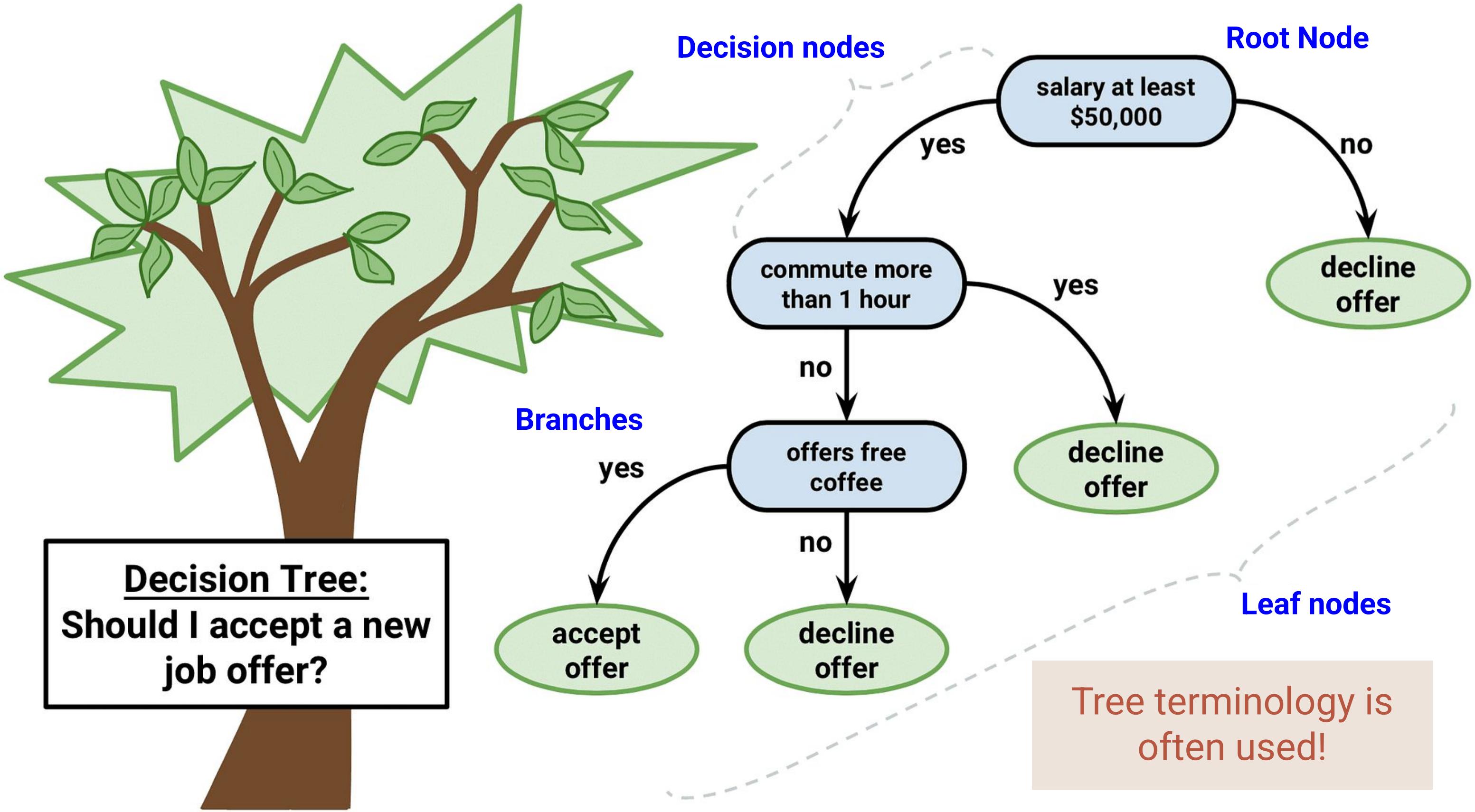
- MyString = `“Hello! My name is Jim\n”`
- Next using MIPS replace substring “Jim” with “Meg”
- Substring = MyString[18:20] = `“Jim”` (byte-index)

```
.data
MyString: .asciiz  "Hello! My name is Tim\n"

.text
        la $a0, MyString
        li $t1, 'M'             # Load character ‘M’ into $t1
        addi $t2, $a0, 18       # Add 18 to the base address of MyString
        sb $t1, 0($t2)          # Store ‘M’ to index 19 of MyString
        li $t1, 'e'             # Load character ‘e’ into $t1
        addi $t2, $a0, 19       # Add 19 to the base address of MyString
        sb $t1, 0($t2)          # Store ‘e’ to index 20 of MyString
        li $t1, 'g'             # Load character ‘g’ into $t1
        addi $t2, $a0, 20       # Add 20 to the base address of MyString
        sb $t1, 0($t2)          # Store ‘g’ to index 21 of MyString
```

This would be so much better in a loop....

Decision Making Operations



Tree terminology is often used!

Decision Making: Conditional Operations

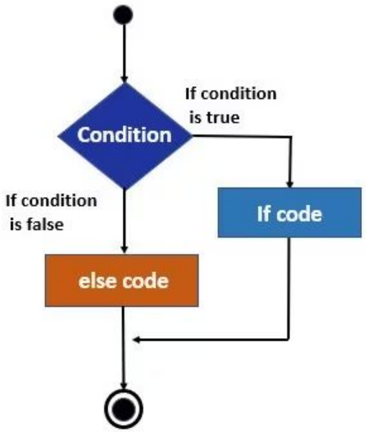
- **Branch** to a labeled instruction if a condition is true
  - Otherwise, continue sequentially
- `beq rs, rt, L1`
  - if (rs == rt) branch to instruction labeled L1
- `bne rs, rt, L1`
  - if (rs != rt) branch to instruction labeled L1
- `j L1`
  - unconditional jump to instruction labeled L1

Used to perform if, while and for loops

Conditional Operations – If-Then-Else

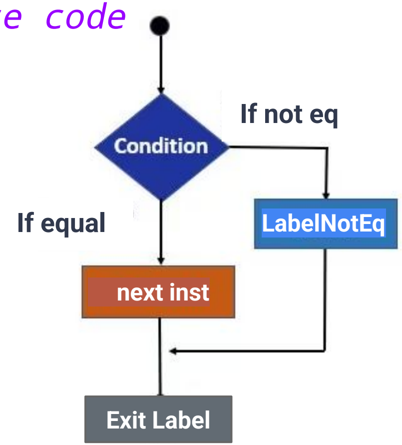
Python example #1

```
if (a != b)
     # code if true
Else:
     # code if false
```



MIPS example#1

```
     bne $s3, $s4, LabelNotEq
     # code if equal
     j   ExitLabel
LabelNotEq: # code if not eq
ExitLabel: # more code
```



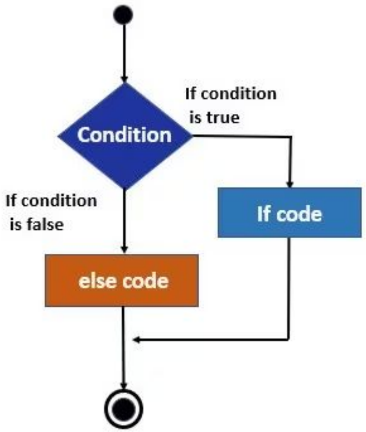
Relevant instructions

- **`beq rs, rt, LabelEq`** = Branch to LabelEq if (rs == rt)
- **`bne rs, rt, LabelNotEq`** = Branch to LabelNotEq if (rs != rt)
- **`j ExitLabel`** = jump unconditionally to ExitLabel

Conditional Operations – If-Then-Else

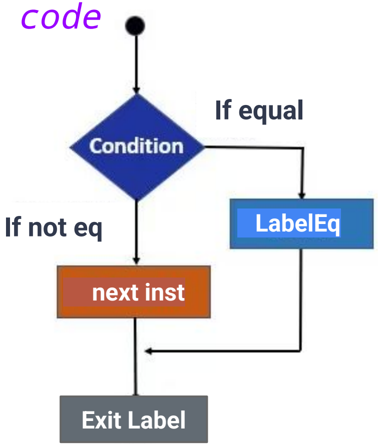
[Python example #2]

```
if (a == b)
     # code if true
Else:
     # code if false
```



[MIPS example#2]

```
     beq $s3, $s4, LabelEq
     # code if not eq
     j   ExitLabel
LabelEq: # code if eq
ExitLabel: # more code
```



Relevant instructions

- **`beq rs, rt, LabelEq`** = Branch to LabelEq if (rs == rt)
- **`bne rs, rt, LabelNotEq`** = Branch to LabelNotEq if (rs != rt)
- **`J ExitLabel`** = jump unconditionally to ExitLabel

If-then-else Example – My Turn

Given the following registers initial value.
Execute the following if-then-else program:

Name	Number	Value
$t0	8	5
$t1	9	0
$t2	10	3
$t3	11	5
$t4	12	3
$t5	13	15
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	75
$s2	18	0

```
       bne $s0, $s1, Then      If (condition)
       addi $t2, $t2, 2        Code if false
       j    Exit
Then:  addi $t2, $t2, -2       Code if true
Exit:
```

What is this code doing?

```
if (____________)
    $t2 = __________
else
    $t2 = __________
```

What is the end result for $t2? ______

More Conditional Operations

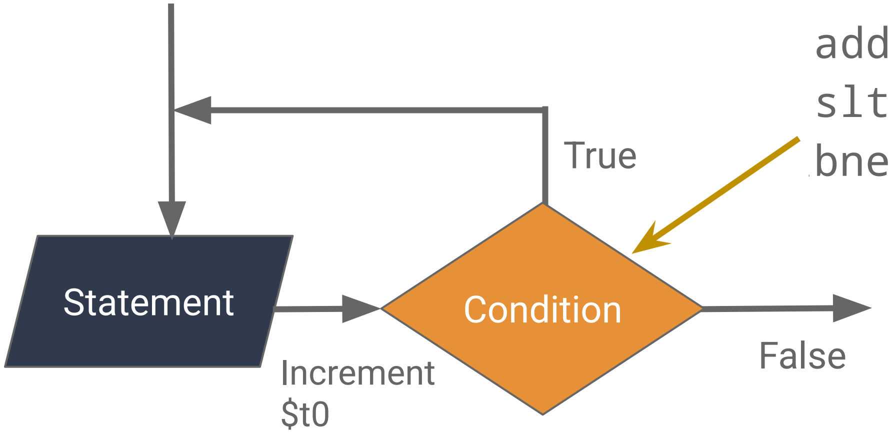
- **Set** result to 1 if a condition is true
  - Otherwise, set to 0
- `slt rd, rs, rt # Set Less Than`
  - if (rs < rt) rd = 1; else rd = 0;
- `slti rt, rs, constant  #Set Less Than Imm`
  - if (rs < constant) rt = 1; else rt = 0;
- Often used in combination with beq, bne for other complex conditions (<, <=, >, >=)

```
slt $t0, $s1, $s2          # if ($s1 < $s2)
bne $t0, $zero, L          #   branch to L
```

Conditional Operations – For Loop

Python example #1

```
For i in range(j)
   # code in for loop
```



MIPS example #1

```
add $t0, $zero, $zero  # initialize $t0
Loop:
    # code in for loop
    addi $t0, $t0, 1        # increment $t0
    slt $t2, $t0, $t1       # $t2=1,if $t0<$t1
    bne $t2, $zero, Loop    # branch if $t2!=0
```

Relevant instructions

- **`slt rd, rs, rt`** → Set rt = 1 (True), if (rs<rt), else rt = 0 (False)
- **`slti rt, rs, Imm`** → Set rt = 1 (True), if (rs<Imm), else rt = 0 (False)
- **`bne rs, rt, ProcLabel`** → Branch to ProcLabel if (rs != rt)

For Loop Example – My Turn

MIPS code:

```
    addi $s1, $zero, 2
    addi $t0, $zero, 3
Loop:
    addi $s1, $s1, 3
    addi $t0, $t0, 1
    slti $t1, $t0, 5
    bne $t1, $zero, Loop
```

Initialization code

Code in for loop

For loop condition

What is the final state of the Register Table?
Let's run the code line-by-line

Name	Number	Value
$t0	8	5
$t1	9	0
$t2	10	3
$t3	11	5
$t4	12	3
$t5	13	15
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	75
$s2	18	0

Loop#	0	1	2	3	4	5
$s1	2					
$t0	3					
$t1	0					
Branch?						

For Loop Example – My Turn

MIPS code:

```
    addi $s1, $zero, 2
    addi $t0, $zero, 3
Loop:
    addi $s1, $s1, 3
    addi $t0, $t0, 1
    slti $t1, $t0, 5
    bne $t1, $zero, Loop
```

What is the final state of the Register Table?
Let's run the code line-by-line

Algorithm
$s1 = 2+2x3 = 8
$t0 = 5, $t1 = 0

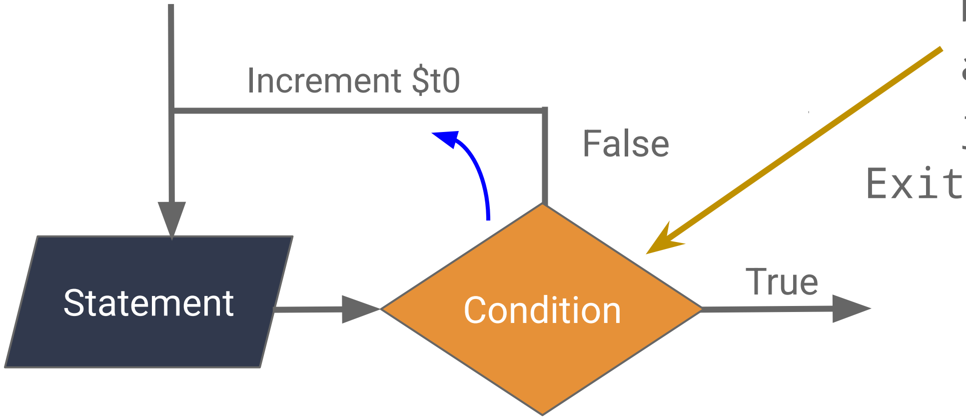
Name	Number	Value
$t0	8	5
$t1	9	0
$t2	10	3
$t3	11	5
$t4	12	3
$t5	13	15
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	8
$s2	18	0

Loop#	0	1	2	3	4	5
$s1	2	5	8			
$t0	3	4	5			
$t1	0	1	0			
Branch?		Y	N			

Conditional Operations – While Loop

Python example

```
i = 1;
while i < 6:
  # code in while loop
  i += 1
```



MIPS example

```
add $t0, $zero, 1  # initialize $t0
add $t2, $zero, 6  # set stop cond.
Loop:
    # code in while loop
    beq $t0, $t2, Exit  # branch $t0=$t2
    addi $t0, $t0, 1    # increment $t0
    j Loop
Exit:
```

Relevant instructions

- **`beq rs, rt, LabelEq`** = Branch to LabelEq if (rs == rt)
- **`bne rs, rt, LabelNotEq`** = Branch to LabelNotEq if (rs != rt)
- **`J ExitLabel =`** jump unconditionally to ExitLabel

While Loop Example – Your Turn

MIPS code:

```
Loop:
      add   $t1, $t1, $t4
      beq   $t0, $t2, Exit
      addi  $t0, $t0, 1
      j     Loop
Exit: …
```

Loop #	0	1	2	3	4	5
$t0	5					
$t1	0					
$t2	9					
Branch?						

What is the final state of the Register Table?

Name	Number	Value
$t0	8	5
$t1	9	0
$t2	10	9
$t3	11	5
$t4	12	3
$t5	13	15
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	75
$s2	18	0

Algorithm $t0 =
$t1 =

Loops for String manipulation

- Strings are simply arrays of characters
- CPU uses for- and while- loops to do string manipulation
  - Concatenation, Substring Extraction, Searching and Indexing, Replacing and Modifying, Splitting and Joining, Transformations and Formatting
- Use MIPS instructions load byte (`lb`) and store byte (`sb`)

Example – String Copy using a while loop

Pseudocode:

```
i = 0;
while ((y[i]=x[i])!='\0')
  i += 1;
```

Example
Input:
x = "Hey you!"
y = ""
Output:
x = "Hey you!"
y = "Hey you!"

Follow along by typing code into MARS. Lecture 6-String copy.asm is in Teams.

```
.data
x: .asciiz "Hey you!"
y: .space 8

.text
la $a0, x                   # Base address of x
la $a1, y                   # Base address of y
add  $s0, $zero, $zero      # i = 0

L1:___________________      # Get address of x[i]
   ___________________      # $t2 = x[i]
   ___________________      # Get address of y[i]
   ___________________      # y[i] = x[i]
   ___________________      # exit loop if x[i]==0
   ___________________      # i = i + 1
   j L1                     # next iteration of loop
Done: ...
```

MIPS pair programming exercise

- Write a MIPS program that executes the following algorithm. Pseudocode is below.
- Work in teams of 2: One person is the "driver" (writes the code), the other is the "navigator" (reviews and guides).

Pseudocode:

```
i = 0;j = 0;
for i from 0 to N-1
  Result[i] = x[i];
for j from 0 to M-1
  Result[N+j] = y[j];
```

Given:

- Base addresses of string x, y in \$a0, \$a1
- N in \$s0, M in \$s1
- i in \$t0, j in \$t1
- Base address of string Result in \$v0

What string manipulation function does this code perform?

Summary

- Arrays: Data and Strings
- Conditional Decision Operations
  - Branch, Set, Jump
- If-the-else, For-loops, While-loops
- Examples

Next lecture

Shifters, Logical, Machine Coding - Part 1