Branching and Memory Access

CS230 Tutorial 04
Branching

Branching checks a condition and branches to the line if the condition is true. Branching can be done with either a line label or an integer (number of lines to skip, which can be negative)

```
beq $s, $t, i
If $s = $t, then skip i lines
bne $s, $t, i
If $s ≠ $t, then skip i lines
```

Remember that this is not the same as an if-else, there is no skipping after a block.
Branching Example - Function Cases

Code the following function, assuming m, n, p refer to contents of registers $1, $2, $3 respectively.

a. \( p = (m + 10) - (n + 20) \)
b. If n is not 0, \( p = m/n \)
Branching Example - Function Cases

addi $3, $1, 10
sub $3, $3, $2
addi $3, $3, -20
beq $2, $0, 2
div $1, $2
mflo $3
jr $31

bne $2, $0, 4
addi $3, $1, 10
sub $3, $3, $2
addi $3, $3, -20
beq $0, $0, 2
div $1, $2
mflo $3
jr $31
Set Less Than

Set less than is used to compare two numbers, it is generally used in conjunction with branching which can only check for equality or nonequality.

```
slt $d, $s, $t
```

If $s < $t then $d = 1
If $s ≥ $t then $d = 0

Be careful to match the order of arguments with the comparison you want!
Branching Example - slt

Code the following function, assuming x, y, z are in registers $1$, $2$, $3$ respectively.

$z = x^y$
Branching Example - slt

\[
\begin{align*}
\text{addi } & \quad $3, \quad $0, \quad 1\\
\text{loop:} & \quad \text{slt } $4, \quad $0, \quad $2\\
& \quad \text{beq } $4, \quad $0, \quad \text{endloop}\\
& \quad \text{mult } $3, \quad $1\\
& \quad \text{mflo } $3\\
& \quad \text{addi } $2, \quad $2, \quad -1\\
& \quad \text{beq } $0, \quad $0, \quad \text{loop}\\
\text{endloop:} & \quad \text{jr } $31\\
\end{align*}
\]

Equivalent to:

\[
\begin{align*}
z = 1\\
\text{while } y > 0, \quad \text{do:}\\
z = z \times x\\
y = y - 1
\end{align*}
\]
Memory Access

lis $d
Store the next line into $d

.word i
Read i as a word rather than a command

lw $t, i($s)
Get memory address in $s and add i, load the value at that address into $t

sw $t, i($s)
Get memory address in $s and add i, save the value in $t into that address
Memory Access - Text Parsing

If the user inputs a letter, output that letter, otherwise output nothing.

Standard input is read from memory address 0xFFFF0004. Standard output is read from memory address 0xFFFF000c.

Input and output is encoded with ASCII.

Upper case letters are from 41-5A (65-90)
Lower case letters are from 61-7A (97-122)