Compile a strcpy procedure (shows how to use strings)

// x and y hold strings
// each element is a char, ∴ only 1 byte long
void strcpy (char x[], char y[])
{
    int i;

    i = 0;
    while ((x[i] = y[i]) != '\0')
        i += 1;
}
x[0] → $a, y[0] → $a1
i → $s0

Adjust the stack pointer and save $s0 (as i), the only register to be used.
strcpy:
    addi $sp, $sp, -4
    sw $s0, 0($sp)

Initialize $s0 to 0 (i = 0)
addi $s0, $zero, $zero

Need to copy y[i] into x[i].

add i to y[0]'s address
L1: add $t1, $s0, $a1

Note: just adding 0 to base address instead of 4 bcos we are dealing with bytes now (each element of x[] is a char of a byte):
Load the character y[i] in $t2
lb $t2, 0($t1)

Similarly load address of x[i] into $t3
add $t3, $s0, $a0
Store the byte from $y[i]$ into $x[i]$:
```
sb $t2, 0($t3)  # x[i] = y[i]
```

Exit if character is 0
```
beq $t2, $zero, L2  # if y[i]=0 go to L2
```

Otherwise, increment $i$ and loop back
```
addi $s0, $s0, 1
j    L1
```

If done, restore $s0
```
L2: lw $s0, 0($sp)
addi $sp, $sp, 4
jr $ra
```

Logical Operations

Shifts

- shift left & right (C ver.: $<<$, $>>$)
  - shifts a register’s value by given amount to left, stores zeros on the right, and stores the new value in new reg.
  - similarly $srl$ (shift right logical)
  - $sll$ $t2$, $s0$, 4    # $t2 = ($s0 shifted left by 4 bits)
  - the $shamt$ in the R-format is used for this

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>16</td>
<td>10</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

- $sll$ encoded by both $op$ and $funct$
- $rd$ $\to$ $t2$
- $rt$ $\to$ $s0$
- $rs$ is unused, so 0
- A bonus
  - shift left by $i$ $==$ multiply by $2^i$

Other Operations

- $and$ $s0$, $t1$, $t2$    # AND of $t1$ and $t2$
- $or$ $s0$, $t1$, $t2$     # OR of $t1$ and $t2$
- $nor$ $s0$, $t1$, $t2$    # NOR of $t1$ and $t2$
- $andi$ $s0$, $t0$, 100    # AND of $t0$ and 100