Branch Encoding

- To encode branch instructions, we first need to calculate the value for the associated label. This is done by the assembler
- Mips has restricted alignment. The encoded instruction are word aligned, which means all the labels will be multiples of 4.
- To increase the range, we encode the address divided by 4
  - In other words, the address is in terms of words/instructions (32 bit), rather than bytes

Branch target/ branch displacement/ branch offset: the number of instructions between the branch instruction and the target instruction

- It should be calculated as the

$$\frac{(targetaddress - currentaddress)}{4}$$

```
l2:     instruction                     # Opcode :4
                instruction                     # rs: t0(8), rt: t1 (9)
                instruction                     # branch target:-4 (16 b)
                BEQ $t1, $t0, L2
```

Mips J Format

- Used for unconditional jumps and function calls
- The opcode field is used to identify the type of instruction
- The targaddr field is used to indicate a pseudo-direct target address
- Pseudo direct addressing is used to provide a maximum range for the jumps while still maintaining a 32-bit address into a 26 bit-field

This is done by...

```
Enter NOTES from 5/31 here
```

Multiply and Divide Instructions

2 Address Forms. These instructions implicitly use the internal registers hi and lo. We would then need to move the values from these internal registers into the programmer-usable registers.

- `mult rs, rt`

    - puts the high word in hi and low word in lo
- `div rs, rt`
    - puts the remainder in hi and the quotient in lo
- `mfhi $rd`
    - Copies the value in hi to the given register
- `mflo $rd`
    - Copies the value int lo to the given register

% is the same then as normal division within MIPS programming

Character and String Operations

- Characters are encoded as 0's and 1's using ASCII encoding
- Each character is represented using 8 bits (or a byte)
- A string can be stored in the data segment by using the .asciiz directive. This would be a C-style string (array of chars)
- MIPS provides instructions to move bytes to work with individual characters
- LB (load byte) loads a byte to the rightmost 8 bits of a register
- SB (store byte) writes the rightmost 8 bits of a register to memory

Sys call 4 - Prints the string with the starting address 0
Sys call 11 - Prints character stored in a0
Sys call 8 - reads a string from console. a0 should contain the starting address of the unformatted free space, and a1 should contain the maximum number of characters to read
Sys call 12 - reads a character from the console and returns it in v0

Functions

`jal l1` - Jump and link so the computer knows where to return to following the jump