Aula 8 - Linguagem de montagem - 08/07

Dica para o exercício de paridade

Fazer um and com uma máscara para contar os bits 1:

```
    li $t0, 1
loop:
    and $t1, $t0, $a0
    beq $t0, $t1, soma
    slt $t0, $t0, 1
    j loop
soma:
    addi $t3, $t3, 1
    j loop
```

Caracteres

- Conjunto especial tabelado
  - Tabela ASCII: mapeia um caractere para um número de 8 bits (1 byte).
  - Tabela Unicode

Instruções para trabalhar com caracteres

Temos as instruções load byte e load byte unsigned:

- lb/lbu reg, offset(base)

Carregam apenas um byte no registrador de 4 bytes. Carregam o número e em seguida fazem a extensão de sinal. O lbu apenas completa com 0 os bits mais significativos.

> Para trabalhar com os caracteres utilizamos o lbu, já que os caracteres não possuem sinal.

Para armazenar um byte, utilizamos a instrução:

- sb reg, offset(base)

Exemplo:

```
void strcpy(char *x, char *y){
    int i = 0;
    do { // Observação: '\0' é o 0 na tabela ASCII.
        x[i] = y[i];
        i++;
    } while (y[i] != '\0');
}
```

```
strcpy:                     # $a0 = *x e $a1 = *y
    add $t0, $zero, $zero   # i = 0
loop:
    add $a1, $a1, $t0       # y = y + i
    lbu $t1, 0($a1)         # t1 = y[i]
    add $a0, $a0, $t0       # x = x + i
    sb $t1, 0($a0)          # x[i] = t1 (= y[i]).
    addi $t0, $zero, 1      # i++; Incremento de 1 pois cada caractere utiliza apenas 1
    bne $t1, $zero, loop    # if y[i] != 0 goto loop
    jr $ra
```

Constantes de 32 bits

- Instruções do formato tipo I:

op	rs	rt	const
6 bits	5 bits	5 bits	16 bits

- O tamanho do registrador é 5 bits pois há 32 registradores (2^5) diferentes.

Como armazenar números maiores do que 16 bits em um registrador?

Exemplo: armazenando 4 milhões em um registrador. | 0000000000111101 | 0000100100000000 | | — | — | | 16 bits | 16 bits | | 61 | 2304 |

```
addi $s0, $zero, 61
sll $s0, $s0, 16
addi $s0, $s0, 2304
```

- Instrução load upper immediate: armazena 16 bits nos bits mais significativos de um registrador.

```
lui $s0, 61     # lui é uma pseudoinstrução.
addi $s0, $s0, 2304
```