## Acesso indexado (load byte e load word):

Exemplo str[i]

Vais precisar de 3 registos:

- i : $t0
- Str : $t1
- Str + i : $t2
- Str[i] : $t3

Str é um array de chars

```
Li $t0, 0                # $t0 = i = 0
La $t1, str              # $t1 = str ou &str[0]
Addu $t2, $t1, $ t0      # $t2 = str + i ou &str[i]
Lb $t3, 0($t2)           #  $t3 = str[i]
```

No final do loop:

```
addi $t1, $t1, 1              # i++;
```

1. **Addu $t2, $t1, $t0**: Adiciona o conteúdo de **$t0** (que é o índice **i**) ao endereço base da string (**$t1**). O resultado, armazenado em **$t2**, será o endereço da posição **str + i** ou, mais especificamente, o endereço do caractere na posição **i** da string, **&str[i]**.
2. **Lb $t3, 0($t2)**: Carrega um byte da memória. Ele utiliza o endereço calculado em **$t2** (que é o endereço da posição **&str[i]**) e coloca o valor contido nessa posição no registrador **$t3**. Assim, **$t3** conterá o valor de **str[i]**.

**Se tivesses um array de inteiros terias de multiplicar o teu i por 4 ou seja:**

```
Li $t0, 0                # $t0 = i = 0
La $t1, array            # $t1 = array ou &array[0]
Mul $t5, $t0, 4          # i * 4;
Addu $t2, $t1, $t5       # array+i;
Lw $t3, 0($t2)           # array[i]
```

No final do loop:

```
addi $t1, $t1, 1         # i++;
```

1. **La $t1, array**: Carrega o endereço do array (**array**) no registrador **$t1**. **$t1** agora contém o endereço base do array ou, em outras palavras, o endereço do primeiro elemento do array.
2. **Mul $t5, $t0, 4**: Multiplica o valor de **$t0** (índice **i**) por 4, pois cada elemento em um array de inteiros geralmente ocupa 4 bytes na memória. O resultado, armazenado em **$t5**, é o deslocamento em bytes a ser adicionado ao endereço base para acessar o elemento desejado.
3. **Addu $t2, $t1, $t5**: Adiciona o conteúdo de **$t5** ao endereço base do array (**$t1**). O resultado, armazenado em **$t2**, é o endereço da posição **array + i * 4**, ou seja, o endereço do elemento **i** no array de inteiros, &str[i].
4. **Lw $t3, 0($t2)**: Carrega uma palavra (4 bytes) da memória. Utiliza o endereço calculado em **$t2** (que é o endereço da posição &**array[i]**) e coloca o valor contido nessa posição no registrador **$t3**. Assim, **$t3** conterá o valor de **array[i]**

**No acesso indexado somas no final do ciclo sempre i +1**

## Ponteiros (load byte e load word):

- p: $t1
- *p : $t2

```
La $t1, str                    #       p = str ou &str[0]
Lb $t2 , 0($t1)                #      *p = str[0]
No final do loop:
addi $t1, $t1, 1               # i++;
```

1. **La $t1, str**: Carrega o endereço da string (**str**) no registrador **$t1**. Aqui, **$t1** passa a conter o endereço base da string ou o endereço do primeiro caractere da string.
2. **Lb $t2, 0($t1)**: Carrega um byte da memória. Utiliza o endereço contido em **$t1** (que é o endereço da posição **str[0]**) e coloca o valor contido nessa posição no registrador **$t2**. Assim, **$t2** conterá o valor de ***p** ou **str[0]**.

**Se tivesses um array de inteiros:**

```
la $t1, array        # $t1 = array ou &array[0]
lw $t2, 0($t1)       # $t2 = *p = array[0]
```

Agora, no final do loop:

```
addi $t1, $t1, 4      # Incrementa o ponteiro para o próximo elemento inteiro (array[i])
```

- **la $t1, array**: Carrega o endereço do array (**array**) no registrador **$t1**. Aqui, **$t1** contém o endereço base do array ou o endereço do primeiro elemento do array.
- **lw $t2, 0($t1)**: Usa **lw** (load word) para carregar uma palavra (4 bytes) da memória. Utiliza o endereço contido em **$t1** (que é o endereço da posição **array[0]**) e coloca o valor contido nessa posição no registrador **$t2**. Assim, **$t2** conterá o valor de ***p** ou **array[0]**.
- **addi $t1, $t1, 4**: Incrementa o ponteiro **$t1** para apontar para o próximo elemento inteiro no array. Isso é feito adicionando 4 ao endereço, pois cada inteiro ocupa 4 bytes.

## Acesso indexado (store word e store byte):

### Com array de ints:

```
la $t1, lista                 #     $t1 = lista;
mul $t3, $t0, 4
addu $t2, $t1, $t3            #     $t2 = lista + i;
li $v0, read_int
syscall
sw $v0, 0($t2)                #     lista[i] = read_int();
addi $t0, $t0, 1
```

1. **la $t1, lista**: Carrega o endereço da lista (**lista**) no registrador **$t1**. Aqui, **$t1** passa a conter o endereço base da lista.
2. **mul $t3, $t0, 4**: Multiplica o valor de **$t0** (que representa o índice **i**) por 4. Isso é feito porque, comumente, em arrays de inteiros, cada elemento ocupa 4 bytes na memória. O resultado, armazenado em **$t3**, é o deslocamento em bytes a ser adicionado ao endereço base para acessar o elemento desejado.
3. **addu $t2, $t1, $t3**: Adiciona o conteúdo de **$t3** ao endereço base da lista (**$t1**). O resultado, armazenado em **$t2**, é o endereço da posição **lista + i * 4**, ou seja, o endereço do elemento **i** na lista de inteiros.

4. **sw $v0, 0($t2)**: Armazena o valor lido da syscall na posição de memória apontada por **$t2**. Em outras palavras, essa linha atribui o valor lido pelo **read_int** à posição **lista[i]**.
5. **addi $t0, $t0, 1**: Incrementa o valor no registrador **$t0** (que representa o índice **i**) em 1. Isso prepara o registrador para o próximo ciclo do loop, se houver um loop envolvendo esse trecho de código.

### Com array de chars:

```
la $t1, lista       # $t1 = lista;
addu $t2, $t1, $t3  # $t2 = lista + i;
li $v0, 12          # Código da syscall para ler um caractere (read_char)
syscall
sb $v0, 0($t2)      # Armazena o caractere lido em lista[i];
addi $t0, $t0, 1    # Incrementa i
```

- **li $v0, 12**: A syscall **12** é usada para ler um caractere (**read_char**). Diferentemente da syscall **read_int** que usamos anteriormente para ler um inteiro.
- **sb $v0, 0($t2)**: Em vez de **sw**, usamos **sb** para armazenar um byte (caractere) na memória. Essa instrução faz sentido quando você está trabalhando com um array de caracteres.

## Ponteiros (store word ou store byte):

### Com array de ints:

```
la $a0, str1
li $v0, print_str
syscall                 #   print_string(str1);
li $v0, read_int
syscall                 #   read_int();
sw $v0, 0($t6)          #   *p = read_int();
addiu $t5, $t5, 4       #   p++;
```

1. **la $a0, str1**: Carrega o endereço da string **str1** no registrador **$a0**. Aqui, **$a0** conterá o endereço base da string.
2. **sw $v0, 0($t6)**: Armazena o valor lido da syscall na posição de memória apontada por **$t6**. Isso é semelhante à operação ***p = read_int();** em C, onde **p** é um ponteiro para algum local de memória.
3. **addiu $t5, $t5, 4**: Adiciona uma unidade ao valor contido em **$t5**. Isso incrementa o ponteiro **p** para que ele aponte para a próxima posição de 4 bytes na memória. O uso de **addiu** é para uma operação de adição imediata sem sinal.

**Com array de chars:**

```
la $a0, str1       # $a0 = endereço base da string str1
li $v0, print_str  # Código da syscall para imprimir uma string
syscall            # Imprime a string str1
li $v0, read_char  # Código da syscall para ler um caractere (read_char)
syscall            # Lê um caractere do usuário e armazena em $v0
sb $v0, 0($t6)     # Armazena o caractere lido na posição de memória apontada por $t6 (*p = read_char)
addiu $t6, $t6, 1  # Incrementa o ponteiro para a próxima posição de 1 byte
```

- **sb $v0, 0($t6)**: Usa **sb** (store byte) para armazenar o caractere lido na posição de memória apontada por **$t6**. Isso é apropriado para um array de caracteres, onde cada elemento ocupa 1 byte.
- **addiu $t6, $t6, 1**: Incrementa o ponteiro **$t6** para a próxima posição de 1 byte.