3.14 - Loops: a utilização de vetores com índice variável

Exemplo: Seja o loop programado na linguagem C:

```
Loop: g = g + A[ i ];
           i = i + j;
    if (i != h) go to Loop;
```

Suponha que A seja um vetor de 100 posições e para g, h, i e j são associados os regs. $s1--$s4. O vetor base está armazenado em $s5.

```
Loop:add $t1, $s3, $s3  # t1 ← 2* i
    add $t1, $t1, $t1    # t1← 4 *i
    add $t1, $t1, $s5   # t1← endereço de A[ i ]
    lw  $t0, 0( $t1)     # t0← A[ i ]

    add $s1, $s1, $t0    # g ← g + A[ i ]
    add $s3, $s3, $s4   # i ← i + j

    bne $s3, $s2, Loop  # desvia para o Loop se i != j
```

3.15 - Loop While:

Exemplo:

```
while (save [ i ] == k
    i = i + j;
```

Suponha que as variáveis estão associadas com os regs. $s3, $s4 e $s5 e que o endereço base do vetor save está em $s6.

```
Loop:add $t1, $s3, $s3   # t1 ← 2* i
    add $t1, $t1, $t1   # t1← 4 *i
    add $t1, $t1, $s6   # t1← endereço de save[ i ]
    lw  $t0, 0( $t1)    # t0← save[ i ]

    bne $t0, $s5, Exit  # desvia para Exit se save[ i ]
!= k
    add $s3, $s3, $s4   # i ← i + j

    j Loop              # desvia para Loop
Exit:
```

3.16 - Compilação do testes Menor do que.

A instrução usada para testar dois valores é conhecida como *set on less than* ou *slt.*
Esta instrução compara valores de dois registradores e atribui o valor 1 a um terceiro se o valor do primeiro registrador for menor que o segundo; caso contrário atribui zero.

```
slt  $t0, $s3, $s4 # Se $s3 < $s4 → $t0 = 1
```

Para criar facilidades de programação um registrador denominado de `$zero` é mapeado no registrador 0.

Exemplo:
Teste se uma variável a associada ao registrador $s0 é menor que b (reg. $s1) e desvie para Less se a condição for verdadeira.

```
slt $t0, $s0, $s1 # Se a < b → $t0 = 1

bne $t0, $zero, Less # desvia para Less se $t0 != 0
                     # isto se a < b
```

Comando Case/Switch:

```
Switch (k) {
        Case 0: f = i + j;
                break;
        Case 1: f = g + h;
                break;
        Case 2: f = g - h;
                break;
        Case 3: f = i - j;
                break;
}
```

(XEROX – PG 69)

3.17 - Suporte a Procedimentos pelo Hardware da Máquina

A idéia de usarmos procedimento em programas se justifica por:

- facilidade de entendimento de código;
- reutilização de código;
- divisão do problema em problemas menores (divisão e conquista).

Durante a execução de um procedimento, o programa que o chamou e o próprio procedimento, precisam executar seis passos:

- Colocar parâmetros em local acessível ao procedimento;
- Transferir o controle para o procedimento;
- Garantir recursos de memória para sua execução;
- Realizar a tarefa;
- Colocar resultado em local acessível ao programa;
- Retornar ao ponto de origem.

Os registradores usados na chamadas de procedimentos são:

- `$a0` → `$a3` – argumentos para passagem de parâmetros;
- `$v0` → `$v1` – retorno de valores do procedimento para o programa.

Ao final de uma chamada ao procedimento é necessário retornar ao ponto de origem.
O MIPS utiliza um registrador que armazena esse ponto de chamada ao procedimento: `$ra (return address)`.
Esse registrador também é conhecido como PC (Program Counter).

O programa chama o procedimento passando os parâmetros armazenados em `$a0` → `$a3` e usa a função `Jump and Link` para desviar para o procedimento X.

```
jal X
```

O procedimento realiza seu processamento, armazenando os seus resultados nos registradores `$v0` → `$v1` e retorna ao programa usando a função de desvio:

```
jr $ra
```

Se o procedimento precisar de mais registradores para passagem ou retorno de parâmetros, ele deverá usar uma estrutura de pilha do tipo LIFO (Last In First Out).

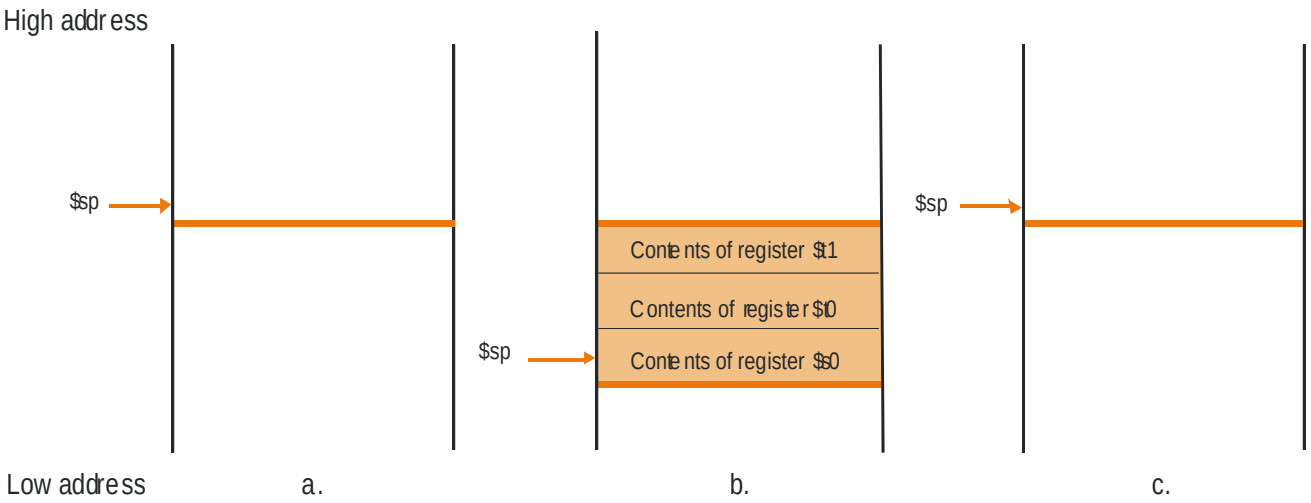
O procedimento que precisar usar a pilha deverá apontar para o endereço do topo da pilha, conhecido como *Stack Pointer*.

As operações na pilha são:

- Push – coloca dados na pilha;
- Pop – retira dados da pilha.

O MIPS reserva um registrador para o *Stack Pointer* `$sp`.

A pilha cresce dos endereços mais altos para os mais baixos, assim ao usar o Push o valor do *Stack Pointer* é decrementado.



Lembrar dos comandos calloc( ), malloc( ) e free( ).

(xérox da página 72 e 77)

Estudar os exemplos de procedimentos aninhados