UNIVALI – Universidade do Vale do Itajaí
PPGC – Mestrado em Computação Aplicada
Disciplina: Arquitetura de Computadores
Professores: Cesar Albenes Zeferino e Douglas Rossi de Melo
Aluno: Fábio Volkmann Coelho
Data:30/05/2025

Avaliação 01 – Programação em linguagem de montagem

Enunciado:
Utilizando a linguagem de montagem do RISC-V, implemente um programa que, considerando um vetor inicializado de 8 posições, determine o maior valor desse vetor e seu índice.

Código-fonte em linguagem de alto nível C:

```c
#include <stdio.h>
#include <limits.h>

int main() {
    int vetor[] = {10, 5, 30, 15, 20, 8, 45, 22};
    int indices = 8;

    int i = 1;
    int maior_valor = vetor[0];
    int k = 0;

    while (i < indices) {
        if (vetor[i] > maior_valor) {
            maior_valor = vetor[i]; // Atualiza o maior valor encontrado.
            k = i;                  // 'k' armazena o índice do maior valor atual.
        }
        i = i + 1; // Incrementa o contador do loop (alternativamente, i++;)
    }
    printf("Maior Valor: %d\n", maior_valor);
    printf("Indice do valor: %d\n", k);
    return 0;
}
```

Execução do Código acima:

```
fabio@fabio-To-Be-Filled-By-O-E-M:~/PycharmProjects/Mestrado/Arquitetura de Computadores/Avaliacao_1$ gcc fabio.c -o executable
fabio@fabio-To-Be-Filled-By-O-E-M:~/PycharmProjects/Mestrado/Arquitetura de Computadores/Avaliacao_1$ ./executable
Maior Valor: 45
Indice do valor: 6
```

Desenvolvimento do código de alto nível:

Antes de começar com a linguagem de montagem, foi elaborado o código em linguagem de alto nível C. Essa etapa inicial teve como objetivo facilitar o entendimento da lógica necessária e, consequentemente, simplificar o processo de desenvolvimento em linguagem de montagem.
Para percorrer o vetor, é possível utilizar estruturas de repetição como while ou for. Ambas executam um trecho de código repetidamente, contanto que uma condição específica seja atendida. Neste projeto, optou-se pelo uso do while. A condição para que o laço continue sua execução é a verificação de que o índice (i) seja menor que o tamanho total do vetor (indices).
Assim, enquanto o índice i for menor que o tamanho do vetor, o programa acessa o valor na posição i desse vetor. Se (IF) esse valor for maior que o conteúdo da variável maior_valor (que armazena o maior número encontrado até aquele momento), então a variável maior_valor é atualizada com esse novo número, e a variável k (que guarda o índice do maior valor) também é atualizada com o valor atual de i. Ao final de cada passagem pelo laço, o valor de i é incrementado, permitindo que o programa avance para o próximo elemento do vetor.
O contador i inicia com o valor 1. Isso se deve ao fato de que o primeiro elemento do vetor (localizado no índice 0) é, inicialmente, considerado como o maior_valor. Dessa forma, seu índice (0) já é armazenado na variável k antes do início do laço.

Código de montagem do RISC-V:

```
.data
vetor: .word 3, 3, 1, 3, 3, 1, 4, 3  # save[0] até save[7]

.text
main:
   la   s6, vetor      # endereço base de save[]
   addi s5, zero, 8    # indices = numero de valores no vetor
   lw   s0, 0(s6)      # maior_valor = primeiro valor é o maior no momento, salvo em s0
   addi s3, zero, 1    # i = inicializa contador do loop
   addi s1, zero, 0    # k = inicializa o indice do maior valor

Loop:
   slli t1, s3, 2          # t1 = i * 4
   add  t1, t1, s6         # t1 = endereço de vetor[i]
   lw   t0, 0(t1)          # t0 = vetor[i]
   ble  t0, s0, continue   # se t0 <= s0, pula atualização do máximo
   add  s0, t0, zero       # atualiza o maior valor (s0 = t0)
   add  s1, s3, zero       # atualiza o indice do maior valor (s1 = s3)

continue:
   addi s3, s3, 1          # i++
   blt  s3, s5, Loop       # se i < tamanho do vetor, continua loop

Exit:
   nop                  # fim do laço (s0 contém o maior valor e s1 contém o valor do indice)
```

Desenvolvimento do código em linguagem de montagem:

Primeiramente, foi preciso definir onde os dados do vetor ficariam guardados na memória. No RISC-V, é utilizada a diretiva .data para indicar que o que vem a seguir são esses dados. Para o vetor, foi utilizado um rótulo (label) chamado vetor. Nele foi armazenado números inteiros, que é definido pela diretiva .word. Finalizada a declaração dos dados as próximas linhas contêm as instruções do programa inciada pela diretiva .text, ou seja, o código de montagem propriamente

dito. O rótulo main é convencionalmente usado para marcar o ponto de entrada, a função principal onde a execução do código se inicia.

Agora, seguindo a lógica que estabelecida no programa em C, o próximo passo foi preparar o acesso aos dados do vetor. Na linguagem de montagem, em vez de carregar o vetor inteiro de uma vez para uma variável, é utilizado o endereço inicial do vetor na memória. Assim, o endereço base do vetor é carregado no registrador s6 (usando a pseudo-instrução la, Load Address). A partir desse endereço base, pode-se calcular a posição de cada um dos outros elementos do vetor somando um deslocamento (offset). Como cada elemento definido com .word ocupa 4 bytes, esse deslocamento é o índice do elemento multiplicado por 4. Após carregar o endereço base em s6, é inicializado o registrador s5 com o tamanho do vetor.

Para carregar o primeiro valor do vetor (vetor[0]) na variável “maior_valor” (que é o registrador s0), foi utilizada a instrução lw (Load Word). Ela lê da memória o valor que está no endereço contido em s6 com um deslocamento de 0 bytes (0(s6)). Com o maior_valor devidamente guardado em s0, carregamos o valor 1 no registrador s3. Este será nosso contador i, e ele começa em 1 porque já o vetor[0] é considerado o maior valor inicial, então a verificação do laço começará a partir do segundo elemento (vetor[1]). Por último, antes de iniciar o laço, o registrador s1 (que representará k, o índice do maior valor) é inicializado com 0, correspondendo ao índice de vetor[0]. Com preparações de dados finalizadas, foi definido um rótulo chamado Loop para marcar onde o laço de repetição começa.

Dentro do Loop, para acessar o elemento vetor[i], é necessário primeiro calcular seu endereço na memória. Isso é feito multiplicando o índice atual i (que está em s3) por 4. No RISC-V, uma forma eficiente de multiplicar por 4 é realizar um deslocamento lógico de 2 bits para a esquerda sobre o valor do índice (instrução slli t1, s3, 2). O resultado (i*4) é armazenado em t1. Em seguida, é somado esse deslocamento (t1) ao endereço base do vetor (s6) para obter o endereço exato do elemento vetor[i]. O valor que está neste endereço calculado é então carregado no registrador t0 usando lw t0, 0(t1).

Por exemplo, na primeira iteração do laço, i (em s3) é 1. O deslocamento calculado será 1 * 4 = 4 bytes. Se o endereço base do vetor (em s6) fosse, hipoteticamente, 0x10010000, o endereço de vetor[1] seria 0x10010004. Se o nosso vetor de exemplo é {3, 3, 1,...}, o valor 3 (que é vetor[1]) seria carregado em t0.

Com vetor[i] em t0, este valor é comparado com o maior_valor atual (que está em s0). Se vetor[i] for menor ou igual ao maior_valor (avaliado pela instrução ble t0, s0, continue), o programa desvia diretamente para o rótulo “continue”, ignorando as instruções de atualização do maior valor.

Caso contrário, se vetor[i] (em t0) for maior que o maior_valor atual (em s0), as atualizações são necessárias: maior_valor (s0) recebe o novo maior valor, que é vetor[i] (de t0), e o índice k (s1) é atualizado com o valor corrente de i (de s3). Após essas atualizações, o fluxo do programa segue para a seção continue (pois não há um desvio que a pule neste caso).

Na seção continue, o contador i (s3) é incrementado em 1 (addi s3, s3, 1). Logo em seguida, o programa verifica a condição de permanência no laço: se i ainda é menor que o tamanho do vetor, ou seja, se s3 < s5 (blt s3, s5, Loop), o programa desvia de volta para o rótulo Loop, iniciando uma nova iteração para verificar o próximo elemento do vetor.

Quando o contador i (s3) não for mais menor que o tamanho do vetor (s5) – ou seja, quando i se tornar igual a s5, a condição do blt falha. Nesse momento, o laço Loop termina. O programa então prossegue para o rótulo Exit (ou simplesmente continua para a próxima instrução após o blt). Isso sinaliza o fim da lógica de busca pelo maior valor. Ao final, o registrador s0 contem o maior valor encontrado no vetor, e s1 contem o índice correspondente a esse valor.