Relatório do Laboratório 05

Alunos: Vinicius Henrique Ribeiro (23200351) e Lucas Furlanetto Pascoali (23204339)
Professor: Marcelo Daniel Berejuck
Disciplina: Organização de Computadores I

Questão 1

A primeira questão solicita a implementação de um procedimento que realizasse o fatorial de um número, atendendo as seguintes premissas:

- Receba via teclado o valor do número a ser calculado o fatorial;
- Efetue o cálculo do fatorial sem o uso de procedimentos;
- Mostre o resultado na tela do computador.

Confira o código abaixo:

```
.data
        quebra_linha:   .asciiz "\n"
.text
        li      $v0, 5
        syscall
        move    $s0, $v0        # salvo o inteiro lido em s0

        mtc1.d  $s0, $f2        # passa o inteiro para o coprocessador 1
        cvt.d.w $f2, $f2        # converte o inteiro para double em f2 para inicializar o acumulador

fatorial:
        addi    $s0, $s0, -1

        mtc1.d  $s0, $f0        # passa o inteiro para o coprocessador 1
        cvt.d.w $f0, $f0        # converte o inteiro para double

        mul.d   $f2, $f2, $f0   # multiplica e acumula
        bgt     $s0, 2, fatorial

        li      $v0, 4
        la      $a0, quebra_linha
        syscall

        li      $v0, 3
        mov.d   $f12, $f2
        syscall
```

Na seção *.data* foi definido apenas uma *string* para realizar a quebra de linha. Já na seção *.text*, o código primeiro realiza a leitura do inteiro fornecido pelo usuário através do teclado e armazena-o no registrador $s0. Devido ao limite de armazenamento de uma *word*, foi utilizado registradores de precisão dupla, permitindo o cálculo de valores maiores, sem que ocorra *overflow*. Com *word*, valores acima de 10 já causavam *overflow*, enquanto os registradores de precisão dupla permitem valores até 200.

Após a leitura do número, movemos o inteiro para o coprocessador 1, mais precisamente no registrador $f2 e convertemos ele para *double*. Após isso, damos

início ao cálculo do fatorial, propriamente. Iniciamos subtraindo 1 do valor armazenado em $s0 (o número fornecido pelo usuário), passamos o inteiro em $s0 para o registrador $f0 e convertemos ele para *double*. Realizamos a multiplicação do número em $f2 (n) pelo número em $f0 (n - 1) e armazenamos o resultado da operação em $f2. Caso o valor em $s0 seja maior que 2, realizamos esse processo novamente. Ao fim do loop, é realizada uma chamada de sistema para imprimir uma quebra de linha na tela e uma outra chamada de sistema que exibe o número em $f2, ou seja, mostra o fatorial do número.

Questão 2

A segunda questão solicita um procedimento que realiza o fatorial de um número de maneira recursiva. O propósito do programa é o mesmo, porém com outra abordagem.

- Receba via teclado o valor do número a ser calculado o fatorial;
- Chame uma função fatorial( ) - procedimento – para calcular, de modo recursivo, o fatorial do número;
- Mostre o resultado na tela do computador

Segue o código:

```
.data
        quebra_linha:   .asciiz "\n"
        um:     .double 1.0
.text
        li      $v0, 5
        syscall

        move    $a0, $v0        # passando argumento para fatorial
        jal     fatorial        # chamanda a função

        li      $v0, 3          # imprimir double
        syscall         # valor já se encontra em f12

        j       exit_main       # encerra o programa


# return a0 * fatorial(a0 - 1)
fatorial:
        ble     $a0, 1, exit_fatorial   # verifica o caso base para encerrar as chamadas recursivas
        addi    $sp, $sp, -8    # move sp para empilhar o contexto atual
        sw      $ra, 4($sp)     # salva o registrador de retorno
        sw      $a0, 0($sp)     # salva o argumento
        addi    $a0, $a0, -1    # configura o argumento para a proxima chamada
        jal     fatorial        # chama a função
        lw      $a0, 0($sp)     # tira o argumenta da pilha
        lw      $ra, 4($sp)     # tira o registrador de retorno da pilha
        addi    $sp, $sp, 8     # move sp para cima

        mtc1.d  $a0, $f2        # passa o inteiro para o coprocessador 1
        cvt.d.w $f2, $f2        # converte o inteiro para double em f2 para multiplicar o retorno

        mul.d   $f12, $f2, $f12 # multiplica o retorno da chamada com o argumento atual
        jr      $ra
exit_fatorial:
        la      $t0, um         # pega o endereço de 1 em double
        l.d     $f12, ($t0)     # põe 1 em f12 para retorno
        jr      $ra             # retorno


exit_main:
```

Em *.data* temos a declaração de uma string para a quebra de linha e de um double com o valor em 1. Já em *.text* temos o código, que carrega para $v0 o código para a leitura de um inteiro, que o armazena em $v0. Movemos o conteúdo de $v0 para $a0 para a passagem como argumento para a função fatorial.

Dentro de fatorial, a primeira instrução é verificar se o argumento é igual ou menor a 1, caso seja, desviamos para o *exit_fatorial* que se encarrega de finalizar o procedimento. Caso o desvio não seja tomado, empilhamos o contexto (regitrador de retorno e o argumento) em $sp, decrementamos o valor do argumento para a chamada recursiva. Este processo ocorre até que a instrução no início do procedimento cause um desvio, fazendo com que todas as instâncias criadas sejam retornadas. Com isso, os procedimentos chamadores continuam seu caminho, ou seja, desempilhando $sp para recuperar o contexto. Com isso, o procedimento tem em $f12 o retorno, em $a0 o argumento e em $ra o endereço de retorno do procedimento chamador. Nas linhas 29 e 30 temos a conversão do *.word* em *.double*, salvo em $f2. Na linha 32 temos a multiplicação, ou seja, valor * (valor - 1) salvo em $f12 (registrador de retorno) e o retorno para o procedimento chamador através de $ra.

A função retorna o fatorial em $f12 para o *main*, que imprime na tela nas linhas 11 e 12.

Questão 3

Testes da questão 1, com BHT entries igual a 16, *BHT History Size* igual a 1 e initial value como NOT TAKEN:

Index	History	Prediction	Correct	Incorrect	Precision
0	NT	NOT TAKE	0	0	0.00
1	NT	NOT TAKE	0	0	0.00
2	NT	NOT TAKE	0	0	0.00
3	NT	NOT TAKE	0	0	0.00
4	NT	NOT TAKE	0	0	0.00
5	NT	NOT TAKE	0	0	0.00
6	NT	NOT TAKE	0	0	0.00
7	NT	NOT TAKE	0	0	0.00
8	NT	NOT TAKE	0	0	0.00
9	NT	NOT TAKE	0	0	0.00
10	NT	NOT TAKE	0	0	0.00
11	NT	NOT TAKE	0	0	0.00
12	NT	NOT TAKE	0	0	0.00
13	NT	NOT TAKE	1	0	100.00
14	NT	NOT TAKE	0	0	0.00
15	NT	NOT TAKE	0	0	0.00

Valor de n = 1;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT	NOT TAKE	0	0	0.00
1	NT	NOT TAKE	0	0	0.00
2	NT	NOT TAKE	0	0	0.00
3	NT	NOT TAKE	0	0	0.00
4	NT	NOT TAKE	0	0	0.00
5	NT	NOT TAKE	0	0	0.00
6	NT	NOT TAKE	0	0	0.00
7	NT	NOT TAKE	0	0	0.00
8	NT	NOT TAKE	0	0	0.00
9	NT	NOT TAKE	0	0	0.00
10	NT	NOT TAKE	0	0	0.00
11	NT	NOT TAKE	0	0	0.00
12	NT	NOT TAKE	0	0	0.00
13	NT	NOT TAKE	1	2	33.33
14	NT	NOT TAKE	0	0	0.00
15	NT	NOT TAKE	0	0	0.00

Valor de n = 5;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT	NOT TAKE	0	0	0.00
1	NT	NOT TAKE	0	0	0.00
2	NT	NOT TAKE	0	0	0.00
3	NT	NOT TAKE	0	0	0.00
4	NT	NOT TAKE	0	0	0.00
5	NT	NOT TAKE	0	0	0.00
6	NT	NOT TAKE	0	0	0.00
7	NT	NOT TAKE	0	0	0.00
8	NT	NOT TAKE	0	0	0.00
9	NT	NOT TAKE	0	0	0.00
10	NT	NOT TAKE	0	0	0.00
11	NT	NOT TAKE	0	0	0.00
12	NT	NOT TAKE	0	0	0.00
13	NT	NOT TAKE	6	2	75.00
14	NT	NOT TAKE	0	0	0.00
15	NT	NOT TAKE	0	0	0.00

Valor de n = 10;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT	NOT TAKE	0	0	0.00
1	NT	NOT TAKE	0	0	0.00
2	NT	NOT TAKE	0	0	0.00
3	NT	NOT TAKE	0	0	0.00
4	NT	NOT TAKE	0	0	0.00
5	NT	NOT TAKE	0	0	0.00
6	NT	NOT TAKE	0	0	0.00
7	NT	NOT TAKE	0	0	0.00
8	NT	NOT TAKE	0	0	0.00
9	NT	NOT TAKE	0	0	0.00
10	NT	NOT TAKE	0	0	0.00
11	NT	NOT TAKE	0	0	0.00
12	NT	NOT TAKE	0	0	0.00
13	NT	NOT TAKE	96	2	97.96
14	NT	NOT TAKE	0	0	0.00
15	NT	NOT TAKE	0	0	0.00

Valor de n = 100;

Com o resultado do BHT de 1 bit com o procedimento fatorial sem usar recursão, podemos perceber que a taxa de erro se mantém em 2, desconsiderando para o valor 1 que tem um acerto sem erros. Isso acontece devido a instrução de desvio sempre ser tomada, menos ao final do cálculo, quando teremos o valor igual a 2. Como começamos com NOT TAKE, considerando valores acima de 2, sempre teremos o erro no início. Ao final nós temos que sair do laço, ou seja, teremos um NOT TAKE mas o histórico irá indicar para um TAKE, resultando então em 2 erros.

Testes da questão 1, com BHT entries igual a 16, *BHT History Size* igual a 2 e initial value como NOT TAKEN:

Index	History	Prediction	Correct	Incorrect	Precision
0	NT, NT	NOT TAKE	0	0	0.00
1	NT, NT	NOT TAKE	0	0	0.00
2	NT, NT	NOT TAKE	0	0	0.00
3	NT, NT	NOT TAKE	0	0	0.00
4	NT, NT	NOT TAKE	0	0	0.00
5	NT, NT	NOT TAKE	0	0	0.00
6	NT, NT	NOT TAKE	0	0	0.00
7	NT, NT	NOT TAKE	0	0	0.00
8	NT, NT	NOT TAKE	0	0	0.00
9	NT, NT	NOT TAKE	0	0	0.00
10	NT, NT	NOT TAKE	0	0	0.00
11	NT, NT	NOT TAKE	0	0	0.00
12	NT, NT	NOT TAKE	0	0	0.00
13	NT, NT	NOT TAKE	2	0	100.00
14	NT, NT	NOT TAKE	0	0	0.00
15	NT, NT	NOT TAKE	0	0	0.00

Valor de n = 1;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT, NT	NOT TAKE	0	0	0.00
1	NT, NT	NOT TAKE	0	0	0.00
2	NT, NT	NOT TAKE	0	0	0.00
3	NT, NT	NOT TAKE	0	0	0.00
4	NT, NT	NOT TAKE	0	0	0.00
5	NT, NT	NOT TAKE	0	0	0.00
6	NT, NT	NOT TAKE	0	0	0.00
7	NT, NT	NOT TAKE	0	0	0.00
8	NT, NT	NOT TAKE	0	0	0.00
9	NT, NT	NOT TAKE	0	0	0.00
10	NT, NT	NOT TAKE	0	0	0.00
11	NT, NT	NOT TAKE	0	0	0.00
12	NT, NT	NOT TAKE	0	0	0.00
13	T, NT	TAKE	0	3	0.00
14	NT, NT	NOT TAKE	0	0	0.00
15	NT, NT	NOT TAKE	0	0	0.00

Valor de n = 5;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT, NT	NOT TAKE	0	0	0.00
1	NT, NT	NOT TAKE	0	0	0.00
2	NT, NT	NOT TAKE	0	0	0.00
3	NT, NT	NOT TAKE	0	0	0.00
4	NT, NT	NOT TAKE	0	0	0.00
5	NT, NT	NOT TAKE	0	0	0.00
6	NT, NT	NOT TAKE	0	0	0.00
7	NT, NT	NOT TAKE	0	0	0.00
8	NT, NT	NOT TAKE	0	0	0.00
9	NT, NT	NOT TAKE	0	0	0.00
10	NT, NT	NOT TAKE	0	0	0.00
11	NT, NT	NOT TAKE	0	0	0.00
12	NT, NT	NOT TAKE	0	0	0.00
13	T, NT	TAKE	5	3	62.50
14	NT, NT	NOT TAKE	0	0	0.00
15	NT, NT	NOT TAKE	0	0	0.00

Valor de n = 10;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT, NT	NOT TAKE	0	0	0.00
1	NT, NT	NOT TAKE	0	0	0.00
2	NT, NT	NOT TAKE	0	0	0.00
3	NT, NT	NOT TAKE	0	0	0.00
4	NT, NT	NOT TAKE	0	0	0.00
5	NT, NT	NOT TAKE	0	0	0.00
6	NT, NT	NOT TAKE	0	0	0.00
7	NT, NT	NOT TAKE	0	0	0.00
8	NT, NT	NOT TAKE	0	0	0.00
9	NT, NT	NOT TAKE	0	0	0.00
10	NT, NT	NOT TAKE	0	0	0.00
11	NT, NT	NOT TAKE	0	0	0.00
12	NT, NT	NOT TAKE	0	0	0.00
13	T, NT	TAKE	95	3	96.94
14	NT, NT	NOT TAKE	0	0	0.00
15	NT, NT	NOT TAKE	0	0	0.00

Valor de n = 100;

Temos uma situação parecida para o BHT com 2 bits com valor inicial NOT TAKE, só que os erros sempre se mantêm constantes em 3. Isso ocorre devido a “resistência” da tabela, pois ela considera 2 bits, fazendo com que ela “insista” no erro (NT, NT) vai para (T, NT), ou seja, depois de um erro prevendo um NOT TAKE, ela tenta um NOT TAKE de novo, errando duas vezes no início e uma no final, quando ele sai do loop.

Testes da questão 2, com BHT entries igual a 16, *BHT History Size* igual a 1 e initial value como NOT TAKEN:

Index	History	Prediction	Correct	Incorrect	Precision
0	NT	NOT TAKE	0	0	0.00
1	NT	NOT TAKE	0	0	0.00
2	NT	NOT TAKE	0	0	0.00
3	NT	NOT TAKE	0	0	0.00
4	NT	NOT TAKE	0	0	0.00
5	NT	NOT TAKE	0	0	0.00
6	NT	NOT TAKE	0	0	0.00
7	NT	NOT TAKE	0	0	0.00
8	NT	NOT TAKE	0	0	0.00
9	T	TAKE	0	1	0.00
10	NT	NOT TAKE	0	0	0.00
11	NT	NOT TAKE	0	0	0.00
12	NT	NOT TAKE	0	0	0.00
13	NT	NOT TAKE	0	0	0.00
14	NT	NOT TAKE	0	0	0.00
15	NT	NOT TAKE	0	0	0.00

Valor de n = 1;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT	NOT TAKE	0	0	0.00
1	NT	NOT TAKE	0	0	0.00
2	NT	NOT TAKE	0	0	0.00
3	NT	NOT TAKE	0	0	0.00
4	NT	NOT TAKE	0	0	0.00
5	NT	NOT TAKE	0	0	0.00
6	NT	NOT TAKE	0	0	0.00
7	NT	NOT TAKE	0	0	0.00
8	NT	NOT TAKE	0	0	0.00
9	T	TAKE	4	1	80.00
10	NT	NOT TAKE	0	0	0.00
11	NT	NOT TAKE	0	0	0.00
12	NT	NOT TAKE	0	0	0.00
13	NT	NOT TAKE	0	0	0.00
14	NT	NOT TAKE	0	0	0.00
15	NT	NOT TAKE	0	0	0.00

Valor de n = 5;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT	NOT TAKE	0	0	0.00
1	NT	NOT TAKE	0	0	0.00
2	NT	NOT TAKE	0	0	0.00
3	NT	NOT TAKE	0	0	0.00
4	NT	NOT TAKE	0	0	0.00
5	NT	NOT TAKE	0	0	0.00
6	NT	NOT TAKE	0	0	0.00
7	NT	NOT TAKE	0	0	0.00
8	NT	NOT TAKE	0	0	0.00
9	T	TAKE	9	1	90.00
10	NT	NOT TAKE	0	0	0.00
11	NT	NOT TAKE	0	0	0.00
12	NT	NOT TAKE	0	0	0.00
13	NT	NOT TAKE	0	0	0.00
14	NT	NOT TAKE	0	0	0.00
15	NT	NOT TAKE	0	0	0.00

Valor de n = 10;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT	NOT TAKE	0	0	0.00
1	NT	NOT TAKE	0	0	0.00
2	NT	NOT TAKE	0	0	0.00
3	NT	NOT TAKE	0	0	0.00
4	NT	NOT TAKE	0	0	0.00
5	NT	NOT TAKE	0	0	0.00
6	NT	NOT TAKE	0	0	0.00
7	NT	NOT TAKE	0	0	0.00
8	NT	NOT TAKE	0	0	0.00
9	T	TAKE	99	1	99.00
10	NT	NOT TAKE	0	0	0.00
11	NT	NOT TAKE	0	0	0.00
12	NT	NOT TAKE	0	0	0.00
13	NT	NOT TAKE	0	0	0.00
14	NT	NOT TAKE	0	0	0.00
15	NT	NOT TAKE	0	0	0.00

Valor de n = 100;

Com o procedimento recursivo usando o BHT de 1 bit com NOT TAKE no valor inicial, temos um erro ao final apenas, tendo em vista que o desvio é tomado apenas no caso final de interrupção das chamadas sucessivas.

Testes da questão 2, com BHT entries igual a 16, *BHT History Size* igual a 2 e initial value como NOT TAKEN:

Index	History	Prediction	Correct	Incorrect	Precision
0	NT, NT	NOT TAKE	0	0	0.00
1	NT, NT	NOT TAKE	0	0	0.00
2	NT, NT	NOT TAKE	0	0	0.00
3	NT, NT	NOT TAKE	0	0	0.00
4	NT, NT	NOT TAKE	0	0	0.00
5	NT, NT	NOT TAKE	0	0	0.00
6	NT, NT	NOT TAKE	0	0	0.00
7	NT, NT	NOT TAKE	0	0	0.00
8	NT, NT	NOT TAKE	0	0	0.00
9	NT, T	NOT TAKE	0	1	0.00
10	NT, NT	NOT TAKE	0	0	0.00
11	NT, NT	NOT TAKE	0	0	0.00
12	NT, NT	NOT TAKE	0	0	0.00
13	NT, NT	NOT TAKE	0	0	0.00
14	NT, NT	NOT TAKE	0	0	0.00
15	NT, NT	NOT TAKE	0	0	0.00

Valor de n = 1;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT, NT	NOT TAKE	0	0	0.00
1	NT, NT	NOT TAKE	0	0	0.00
2	NT, NT	NOT TAKE	0	0	0.00
3	NT, NT	NOT TAKE	0	0	0.00
4	NT, NT	NOT TAKE	0	0	0.00
5	NT, NT	NOT TAKE	0	0	0.00
6	NT, NT	NOT TAKE	0	0	0.00
7	NT, NT	NOT TAKE	0	0	0.00
8	NT, NT	NOT TAKE	0	0	0.00
9	NT, T	NOT TAKE	4	1	80.00
10	NT, NT	NOT TAKE	0	0	0.00
11	NT, NT	NOT TAKE	0	0	0.00
12	NT, NT	NOT TAKE	0	0	0.00
13	NT, NT	NOT TAKE	0	0	0.00
14	NT, NT	NOT TAKE	0	0	0.00
15	NT, NT	NOT TAKE	0	0	0.00

Valor de n = 5;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT, NT	NOT TAKE	0	0	0.00
1	NT, NT	NOT TAKE	0	0	0.00
2	NT, NT	NOT TAKE	0	0	0.00
3	NT, NT	NOT TAKE	0	0	0.00
4	NT, NT	NOT TAKE	0	0	0.00
5	NT, NT	NOT TAKE	0	0	0.00
6	NT, NT	NOT TAKE	0	0	0.00
7	NT, NT	NOT TAKE	0	0	0.00
8	NT, NT	NOT TAKE	0	0	0.00
9	NT, T	NOT TAKE	9	1	90.00
10	NT, NT	NOT TAKE	0	0	0.00
11	NT, NT	NOT TAKE	0	0	0.00
12	NT, NT	NOT TAKE	0	0	0.00
13	NT, NT	NOT TAKE	0	0	0.00
14	NT, NT	NOT TAKE	0	0	0.00
15	NT, NT	NOT TAKE	0	0	0.00

Valor de n = 10;

Index	History	Prediction	Correct	Incorrect	Precision
0	NT, NT	NOT TAKE	0	0	0.00
1	NT, NT	NOT TAKE	0	0	0.00
2	NT, NT	NOT TAKE	0	0	0.00
3	NT, NT	NOT TAKE	0	0	0.00
4	NT, NT	NOT TAKE	0	0	0.00
5	NT, NT	NOT TAKE	0	0	0.00
6	NT, NT	NOT TAKE	0	0	0.00
7	NT, NT	NOT TAKE	0	0	0.00
8	NT, NT	NOT TAKE	0	0	0.00
9	NT, T	NOT TAKE	99	1	99.00
10	NT, NT	NOT TAKE	0	0	0.00
11	NT, NT	NOT TAKE	0	0	0.00
12	NT, NT	NOT TAKE	0	0	0.00
13	NT, NT	NOT TAKE	0	0	0.00
14	NT, NT	NOT TAKE	0	0	0.00
15	NT, NT	NOT TAKE	0	0	0.00

Valor de n = 100;

O resultado com BHT de 2 bits é idêntico ao BHT de 1 bit, pois temos apenas um desvio para encerrar as chamadas sucessivas.

Temos que ressaltar que, como os programas têm apenas uma instrução de desvio, não faz diferença a quantidade de linhas da nossa tabela (table entries).

A conclusão é que o programa que possui apenas uma instrução que promove um desvio tomado em todas as iterações, considerando o primeiro desvio e o valor inicial (TAKE ou NOT TAKE) congruentes, temos apenas um erro, que é o desvio que sai do loop (ou indica o valor base do procedimento concorrente).