Arquitetura de Computadores
Lista de Exercícios
05/03/2025

1. Cite os cinco componentes do modelo de von Neumann. Para cada componente, explique sua finalidade.
2. Suponha que uma instrução de 32 bits tenha o formato descrito abaixo, onde DR e SR identificam registradores. Caso haja 60 Opcodes e 32 registradores, qual é a faixa de valores que pode ser representada pelo imediato IMM? Assuma que IMM é um valor em complemento de 2.

Opcode	DR	SR	IMM

3. Considere uma memória endereçável por bytes composta por 512 endereços, numa arquitetura RISC fictícia onde cada instrução ocupa uma palavra de 16 bits.
    a. Quantos bits são necessários para o endereçamento?
    b. Sabendo que os endereços de palavras são alinhados, para implementar uma instrução de desvio que emprega modo de endereçamento PC-relativo com até 20 palavras de distância, quantos bits são necessários para o offset?
    c. Se uma instrução de desvio estiver na localização 6 usando modo de endereçamento PC-relativo, qual o offset para o endereço 10?
4. O design da arquitetura MIPS é calcado nos seguintes princípios: (1) simplicidade favorece a regularidade, (2) tornar casos comuns rápidos, (3) menor é mais rápido e (4) um bom design exige bons compromissos. Dê exemplos práticos de aplicação de cada um dos princípios mencionados.
5. A arquitetura MIPS estudada tem um conjunto de 32 registradores de 32 bits, usados para armazenar valores temporários durante a execução de instruções. Uma alternativa seria projetar uma arquitetura de computador sem um conjunto de registradores, que usa uma pilha na memória para armazenar valores temporários. Nessas condições, descreva suscintamente como implementar as instruções ADD, SUB, MUL, DIV, LOAD, STORE, CALL (para chamar uma função) e RETURN (para retornar de uma função). Quais as vantagens e desvantagens dessa arquitetura em relação à arquitetura MIPS?
6. Considerando a arquitetura MIPS estudada, responda:
    a. Qual é o endereço da palavra de memória 15?
    b. Quais são os endereços dos bytes que compõe a palavra de memória 15?
    c. Desenhe como o número 0xFF223344 fica armazenado na palavra 15 em máquinas big-endian e little-endian. Deixe claro o endereço de cada byte onde os valores são armazenados.
7. Explique como o trecho de programa a seguir pode ser usado para determinar se um computador é big-endian ou little-endian:

```
1.  li $t0, 0xABCD9876
2.  sw $t0, 100($0)
3.  lb $s5, 101($0)
```

8. Escreva as seguintes strings usando codificação ASCII. Escreva suas respostas finais em hexadecimal.
   a. Cool!
   b. Bom dia a todos!
9. Mostre como as strings do exercício 8 são armazenadas em uma memória endereçável por byte em (a) uma máquina big-endian e (b) uma máquina little-endian começando no endereço de memória 0x1000100C. Indique claramente o endereço de memória de cada byte em cada máquina.
10. Converta as seguintes instruções assembly MIPS para linguagem de máquina. Escreva o resultado final em hexadecimal. Indique o tipo de cada instrução.

```
1.  add $t0, $s0, $s1
2.  lw $t0, 0x20($t7)
3.  addi $s0, $0,-10
4.  addi $s0, $0, 73
5.  sw $t1,-7($t2)
6.  sub $t1, $s7, $s2
```

11. Para cada instrução Tipo-I do exercício 10, faça a extensão do imediato de 16 bits para 32 bits, tal como demandado pela instrução.
12. Converta o programa a seguir, escrito em linguagem de máquina, para linguagem assembly MIPS. Os números à esquerda são os endereços das instruções na memória, e os números à direita são a instrução naquele endereço. Posteriormente, faça engenharia reversa de um programa de alto nível que seria compilado nesse programa assembly. Explique com palavras o que o programa faz. $a0 é a entrada e inicialmente contém um número positivo, n. $v0 é a saída.

```
0x00400000 0x20080000
0x00400004 0x20090001
0x00400008 0x0089502A
0x0040000C 0x15400003
0x00400010 0x01094020
0x00400014 0x21290002
0x00400018 0x08100002
0x0040001C 0x01001020
0x00400020 0x03E00008
```

13. A instrução nori não faz parte do conjunto de instruções MIPS, porque a mesma funcionalidade pode ser implementada usando instruções existentes. Escreva um pequeno trecho de código assembly que tenha a seguinte funcionalidade: $t0 = $t1 NOR 0xF234. Use o mínimo de instruções possível.

14. Implemente os seguintes trechos de código de alto nível usando a instrução `slt`. Assuma que as variáveis inteiras g e h estão nos registradores $s0 e $s1, respectivamente.

```
(a) if (g > h)
      g = g + h;
    else
      g = g- h;
(b) if (g >= h)
      g = g + 1;
    else
      h = h- 1;
(c) if (g <= h)
      g = 0;
    else
      h = 0;
```

15. Escreva uma função em linguagem C com o protótipo `int find42(int array[], int size)`. O argumento `size` especifica o número de elementos do array, e `array` especifica o endereço base do array. A função deve retornar o índice da primeira entrada do array que contém o valor 42. Se nenhuma entrada do array for 42, ela deve retornar o valor–1. Em seguida converta a função para assembly MIPS.
16. A função de alto nível `strcpy` copia a sequência de caracteres `src` para a sequência de caracteres `dst`.

```
// C code
void strcpy(char dst[], char src[]) {
  int i= 0;
  do {
    dst[i]= src[i];
  } while (src[i++]);
}
```

    a) Implemente a função `strcpy` em assembly MIPS. Use $s0 para i.
    b) Desenhe a pilha antes, durante e depois da chamada da função `strcpy`. Assuma $sp= 0x7FFFFF00 logo antes de `strcpy` ser chamada.

17. Escreva uma função chamada `fib` em uma linguagem de alto nível que retorne o número de Fibonacci para qualquer valor não negativo `n`. Dica: Você provavelmente vai usar um loop. Comente claramente seu código. Em seguida converta a função `fib` em código assembly MIPS. Adicione comentários após cada linha de código que expliquem claramente o que ele faz. Use o simulador MARS para testar seu código com `fib(9)`.
18. Converta as instruções assembly MIPS `beq`, `j` e `jal` em código de máquina. Os endereços de instrução são fornecidos à esquerda de cada instrução.

(a)

```
0x00401000          beq $t0, $s1, Loop
0x00401004           . . .
0x00401008           . . .
0x0040100C   Loop:  . . .
```

(b)

```
0x00401000           beq $t7, $s4, done
 . . .               . . .
0x00402040   done:  . . .
```

(c)

```
0x0040310C   back:  . . .
 . . .               . . .
0x00405000           beq $t9, $s7, back
```

(d)

```
0x00403000           jal func
 . . .               . . .
0x0041147C   func:  . . .
```

(e)

```
0x00403004   back:  . . .
 . . .               . . .
0x0040400C   j       back
```

19. Considere o seguinte trecho de código em linguagem assembly MIPS. Os números à esquerda de cada instrução indicam o endereço da instrução.

```
0x00400028 add $a0, $a1, $0
0x0040002C jal f2
0x00400030 f1: jr $ra
0x00400034 f2: sw $s0, 0($s2)
0x00400038 bne $a0, $0, else
0x0040003C j f1
0x00400040 else: addi $a0, $a0,-1
0x00400044 j f2
```

   (a) Traduza a sequência de instruções em código de máquina. Escreva as instruções em hexadecimal.
   (b) Liste o modo de endereçamento usado em cada linha de código.

20. Qual é o intervalo de endereços de instruções para os quais as instruções beq e bne podem desviar no MIPS? Dê sua resposta em número de instruções relativas à instrução de desvio condicional.
21. Explique o modo de endereçamento pseudo-direto empregado pelas instruções j e jal. Quais são suas vantagens?
22. Escreva código assembly que salte para uma instrução 64 Minstructions da primeira instrução. Lembre-se de que 1 Minstruction= 2^{20} instruções=1.048.576 instruções. Suponha que seu código comece no endereço 0x00400000. Use um número mínimo de instruções.
23. Considerando o mapa de memória apresentado em aula, ostre como o seguinte programa MIPS seria carregado na memória e executado.

```
# MIPS assembly code
main:
addi $sp, $sp,-4
sw $ra, 0($sp)
lw $a0, x
lw $a1, y
jal diff
lw $ra, 0($sp)
addi $sp, $sp, 4
jr $ra
diff:
sub $v0, $a0, $a1
jr $ra
```

a) Mostre o endereço de cada instrução, colocando-o ao lado de cada instrução assembly.

b) Desenhe a tabela de símbolos mostrando os labels e seus endereços.
c) Converta todas as instruções em código de máquina.
d) Quão grandes (quantos bytes) são os segmentos de dados e texto?
e) Esboce o mapa de memória mostrando onde os dados e as instruções são armazenados.

24. O que são pseudo-instruções. Dê três exemplos e explique como são implementadas.
25. Como a pseudo-instrução `beq $t1,` $imm_{31:0}$`, L` pode ser implementada?