Gabarito - Lista 1 de OCS Teoria Capítulos 1 e 2

Profa. Virgínia Fernandes Mota

Feito em sala no dia 21/03/2016

1 - Discorra sobre os princípios de projeto. Quais são suas relações com a Lei de Moore e a Lei de Amdahl?

São 4 princípios de projeto:

1) Simplicidade favorece a regularidade;

2) Menor significa mais rápido;

3) Agilize os casos mais comuns;

4) Bom projeto exige bons compromissos.

A Lei de Amdahl é usada para encontrar a máxima melhora esperada para um sistema em geral quando apenas uma única parte do mesmo é melhorada. Isto é frequentemente usado em computação paralela para prever o máximo speedup teórico usando múltiplos processadores. Dessa forma, podemos perceber que ela se relaciona diretamente com o terceiro princípio: Agilize os casos mais comuns.

A Lei de Moore surgiu em 1965 através de um conceito estabelecido por Gordon Earl Moore. Tal lei dizia que o poder de processamento dos computadores (entenda computadores como a informática geral, não os computadores domésticos) dobraria a cada 18 meses. Apesar de podermos fazer uma analogia com o tamanho dos transistores e que menor significa mais rápido, não existe uma relação direta entre os princípios de projeto e esta lei.

2 - Justifique a codificação (formato e largura dos campos) das instruções do MIPS.

Por que tudo no MIPS tem 32? Em respeito aos princípios de projeto, podemos manter os princípio de simplicidade e regularidade.

Temos 3 formatos de instrução no MIPS:

Formato Tipo R: 6 5 5 5 5 6. (op rs rt rd shift funct) add, sub, mv...

Formato Tipo I: 6 5 5 16. (op rs rt const) addi, subi, lw, sw, slt....

Formato Tipo J: 6 26. (op end) j

op = indica a operação

3 - Descreva todos os modos de endereçamento suportados pela linguagem de montagem do processador MIPS.

Existem 5 modos de endereçamento:

1) Tipo registrador: representa o dado de acordo com o nome do registrador

2) Tipo base deslocamento: para trabalhar com offsets (por exemplo arrays)

3) Tipo imediato: representa o dado com o valor dele.

4) Tipo relativo ao PC: relativo à próxima instrução que será executada (jal, $ra: chamadas de procedimento)

5) Tipo absoluto (pseudo-direto): usa o endereço absoluto da instrução (j label)

4 - Descreva o mecanismo de suporte a subrotinas/funções no conjunto de instruções do MIPS. Dê um exemplo de uso indicando o estado da pilha antes da chamada, durante a execução e após o retorno da subrotina/função.

Seis etapas devem ser seguidas:

1) Colocar os parâmetros que serão usados ($a0-$a3): Mais parâmetros serão alocados na memória

2) Transferir o controle para o procedimento: jal (jump and link)

3) Adquirir os recursos necessários para o procedimento.

4) Realizar a tarefa desejada.
5) Colocar o valor de retorno: $v0-$v1
6) Retorna o controle para o ponto de origem: jr $ra
jal funct
** _
_
_
_
funct: _
_
jr $ra #volta para **
Sobre a pilha: Vejam o slide 13, aula 2 parte II.

5 - Considerando os padrões de bits abaixo, mostre o que eles representam em uma instrução do MIPS.?

Campos	op	rs	rt	rd	shamt	funct
Binário	101011	01000	10000	00000	00000	000010
Decimal Tipo R	43	8	16	0	0	2
Decimal Tipo I	43	8	16	2		
Decimal Tipo J	43	17825794				

Logo, 1010 1101 0001 0000 0000 0000 0000 0010 é uma instrução do tipo store word:
sw $s0, 2($t0) # Porém, não é possível andar de 2 em 2 bits!! O correto seria o endereço ser múltiplo de 4.

Campos	op	rs	rt	rd	shamt	funct
Binário	001001	00100	10010	01001	00100	100100
Decimal Tipo R	9	4	18	0	9	36
Decimal Tipo I	9	4	18	18724		
Decimal Tipo J	9	9586980				

Logo, 0010 0100 1001 0010 0100 1001 0010 0100 é uma instrução de um tipo que ainda não vimos! E agora?
Esse tipo é o Add Immediate Unsigned, uma instrução do tipo I: addiu $s2,$a0,18724

6 - Quais as diferenças entre memória volátil e memória não-volátil?

Vamos mostrar as diferenças explicando alguns tipos dessas memórias.
RAM (Randon Access Memory)
Memória de acesso aleatório onde são armazenados dados em tempo de processamento, isto é, enquanto o computador estiver ligado, e também todas as informações que estiverem sendo executadas, pois essa memória é mantida por pulsos elétricos. Todo conteúdo dela é apagado ao desligar-se a máquina, por isso é chamada de volátil.
A memória RAM ganhou melhor desempenho trazendo versões mais poderosas, como a DRAM (Dynamic RAM), ou RAM dinâmica, a EDO RAM (Extended Data Out), ou Saída Estendida de Dados, que proporciona um aumento de desempenho de 10% a 30% em comparação com a RAM tradicional, entre outras.
ROM (Read Only Memory)

Memória não volátil, ou seja, somente de leitura, pois a informação que vem gravada nela não pode ser apagada. Nesta vem as características do fabricante e um programa chamado BIOS*, que comanda todas as operações de Entrada e Saída de dados no microcomputador.

A ROM é permanente e não perde seus dados ao desligar o computador.

*BIOS (Basic Input Output System): A função do BIOS é comunicação, ele permite ao microprocessador comunicar-se com outras partes do computador tal como, o vídeo, impressora, teclado, entre outros. Contém informações que foram gravadas pelo fabricante do micro, estão permanentemente gravadas e não podem ser alteradas. Quando ligamos o micro, é o BIOS que o inicia, checando os periféricos que estão ligados a ele.

7 - Mostre o código MIPS necessário para implementar o seguinte comando C:

```
int x[20], y[8];

a = x[10] + x[ y[3] ];
```

lw $t0, 40($s1) # x[10]
lw $t1, 12($s2) # y[3]
sll $t1, $t1, 2 # y[3] * 4
add $t1, $t1, $s1 # base ($s1) + deslocamento ($t1=y[3]*4)
lw $t2, 0($t1) # x[y[3]]
add $s0, $t0, $t2 # x[10] + x[y[3]]; a = $s0

8 - Problema do Fatorial.

//Em C

```
int fact(int n){
    int i, res=n;
    for(i=(n-1);i>1;i--){
        res*=i;
    }
    return res;
}
```

//Em ASSEMBLY
j main
fact:
addi $sp, $sp, -8
sw $s0, 0($sp)
sw $s1, 4($sp)
addi $s0, $a0, -1
add $s1, $a0, $zero
L1: slti $t0, $s0, 2
beq $t0, 1, Exit
mul $s1, $s1, $s0
addi $s0, $s0, -1
j L1
Exit: add $v0, $s1, $zero
lw $s0, 0($sp)
lw $s1, 4($sp)
addi $sp, $sp, 8
jr $ra
main:

addi $a0, $zero, 5
jal fact
//Em C

```c
int fact(int n){
    if(n<2) return 1;
    else return(n * fact(n-1));
}
```

//Em ASSEMBLY
j main
fact:
addi $sp, $sp, -8
sw $a0, 0($sp)
sw $ra, 4($sp)
slti $t0, $a0, 2
beq $t0, 1, Exit
addi $a0, $a0, -1
jal fact
lw $a0, 0($sp)
lw $ra, 4($sp)
addi $sp, $sp, 8
mul $v0, $v0, $a0
jr $ra
Exit: addi $v0, $v0, 1
jr $ra
main: addi $a0, $a0, 5
jal fact

9 - Arquiteturas RISC vs Arquitetura CISC

Um texto bastante interessante sobre as duas arquiteturas pode ser encontrado em:
http://www.gruponetcampos.com.br/2011/03/arquitetura-cisc-e-risc-qual-diferenca/