Arquitectura de Computadores

Licenciatura em Engenharia Informática

Teste 1 (A) – 2010/03/17 – Duração: 2h00m

Nome:_____ Número:_____

Teste sem consulta.
A interpretação do enunciado faz parte da avaliação. Explicite nas suas respostas todas as hipóteses assumidas.
Por favor, tente ser conciso nas sua respostas para que estas caibam na zona delimitada. Em caso de necessidade pode usar o verso da última folha.

1 Teórica - Escolha Múltipla

Indique o número da resposta que achar correcta no quadrado definido para o efeito. Cada resposta errada desconta 25% da cotação da pergunta. A cotação mínima nesta parte é 0.0 valores.

Q-1 [0.5 val.] Numa memória endereçada ao byte

1. cada byte tem 16 bits
2. um endereço é um inteiro a 8 bits (1 byte) sem sinal
3. cada célula de memória contém um byte
4. cada célula de memória contém um inteiro com 4 bytes

Opção correcta ☐

Q-2 [0.5 val.] Uma linguagem de alto-nível tem a vantagem, sobre o assembly, de

1. poder ser compilada para diferentes códigos máquina
2. poder endereçar memória
3. ser mais próxima da máquina alvo
4. permitir implementar ciclos

Opção correcta ☐

Q-3 [0.5 val.] Considere um CPU com endereços de 32 bits e que transfere palavras de 32 bits entre o CPU e a memória. Internamente, o CPU contém os registos MAR (Memory Address Register), MDR (Memory Data Register) e controla o acesso à memória através dos sinais de controlo RD e WR. Numa instrução máquina que guarda o valor V na posição de memória com endereço A:

1. o MAR contém V, MDR contém A e o sinal de controlo RD está activo
2. o MAR contém V, MDR contém A e o sinal de controlo WR está activo
3. o MAR contém A, MDR contém V e o sinal de controlo WR está activo
4. o MAR contém A, MDR contém V e o sinal de controlo RD está activo

Opção correcta ☐

Q-4 [0.5 val.] Contrariamente ao que acontece nos *buses* síncronos, num *bus* assíncrono não é preciso introduzir ciclos de espera porque

1. o *bus* assíncrono é mais rápido
2. o *bus* assíncrono torna os escravos mais rápidos
3. num *bus* assíncrono o escravo informa o mestre da conclusão do tratamento do pedido
4. num *bus* assíncrono o mestre sabe de antemão quando é que o tratamento do pedido termina

Opção correcta ☐

Q-5 [0.5 val.] A semântica da instrução que realiza a chamada a uma subrotina consiste, no geral, em

1. guardar o endereço da instrução imediatamente a seguir à chamada e alterar o PC para conter o endereço onde começa a subrotina
2. guardar o valor corrente do PC e alterá-lo para conter o endereço da instrução imediatamente a seguir à chamada
3. guardar o endereço onde começa a subrotina e alterar o PC para conter o endereço da instrução imediatamente a seguir à chamada
4. apenas alterar o PC para conter o endereço onde começa a subrotina

Opção correcta ☐

Q-6 [0.5 val.] O processador Pentium:

1. É uma máquina de pilha
2. É uma máquina de registos
3. Oferece um conjunto de instruções RISC
4. É um processador de 16 bits

Opção correcta ☐

Q-7 [0.5 val.] O conceito de pipelining permite

1. Que cada instrução demore menos tempo a executar
2. Diminuir o tempo de acesso a memória
3. Aumentar o número de instruções executadas por unidade de tempo
4. Aceder a zonas de memória próximas das que foram acedidas recentemente

Opção correcta ☐

Q-8 [0.5 val.] No processador Pentium o registo EIP contém:

1. A próxima instrução a executar
2. Os dados da próxima instrução a executar
3. O endereço dos dados da próxima instrução a executar
4. O endereço da próxima instrução a executar

Opção correcta ☐

2 Teórica - Não de escolha Múltipla

Q-9 [1.5 val.] Assinale com uma cruz qual das seguintes características pode associar a arquitecturas RISC e/ou CISC.

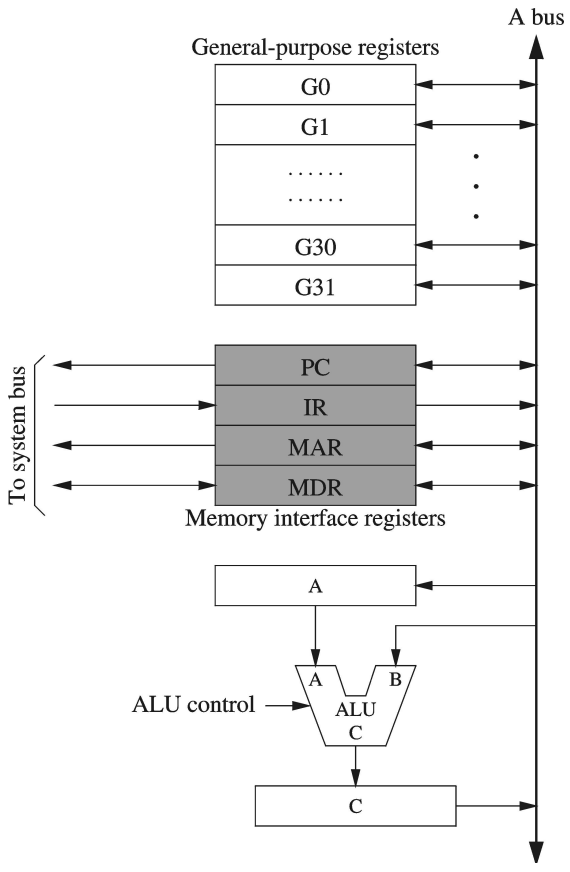
	CISC	RISC
Instruções máquina de tamanho variável		
Endereçar memória em operações aritméticas e lógicas		
Realizar operações aritméticas e lógicas sobre registos		
Utilizar, sempre que possível, registos para a passagem de parâmetros para subrotinas		
Instruções para realizar as operações de LOAD e STORE		

Q-10 [2.0 val.] Quando o CPU quer transferir informação de/para um dispositivo de entrada/saída faz acesso a registos do controlador associado a esse dispositivo. Esses controladores têm um conjunto de endereços normalmente contíguos, por exemplo de 0x400 a 0x407. Estes endereços podem

- pertencer a um espaço de endereçamento completamente disjunto do da memória, ou
- Existir um espaço de endereçamento único: nesse espaço existem endereços que são de células de memória e outros que correspondem a registos de controladores.

Para cada um dos casos anteriores, diga, justificando, que instruções máquina são usadas para transferir informação entre o CPU e os controladores.

Q-11 Considere a seguinte figura que descreve a organização interna de um CPU



Admita que os sinais G0in, G0out, G1in, G1out, ..., G30in, G30out, ..., G31in, G31out, Ain, Aout, Cin, Cout fazem o controlo dos registos e que o controlo da ALU é denotado da seguinta forma: *ALU=op*, sendo *op* operação a executar.
Considere a execução da instrução máquina:
ADDA G2 com a semântica G0 = G0 + G2

a) [1.5 val.] Diga quantos ciclos de relógio são necessários na fase de execução da instrução (note que apenas se pretende a fase de execução da instrução, não interessando aqui a fase de obtenção e descodificação da instrução). Indique o que é feito em cada um dos ciclos.

b) [1.0 val.] Em cada um dos ciclos, indique os sinais de controlo que estão activos.

3 Prática

Q-12 [1.5 val.] Sabendo que numa máquina os endereços ocupam 64 bits, os inteiros 32 bits, e os longs 64 bits, indique:

```
int v1[20];
unsigned long v2[20];
int *pv1 = v1;
unsigned long *pv2 = v2;
sizeof (pv1)  = ____________________;
sizeof (pv2)  = ____________________;
sizeof (*pv2) = ____________________;
sizeof (v1)   = ____________________;
sizeof (v2)   = ____________________;
```

Q-13 [1.5 val.] Considere o seguinte programa (com erros) onde se pretende ler uma palavra, copiar para um novo vector e depois escrever essa palavra, seguida da sua dimensão (número de caracteres).

```
#include <stdio.h>                                  ____________________________________
#include <string.h>                                 ____________________________________

int main( ) {                                       ____________________________________
  char str1[100];                                   ____________________________________
  char *str2;                                       ____________________________________

  scanf( "%s", &str1 );                             ____________________________________
  strcpy( str2, str1 );                             ____________________________________
  printf( "string: %s, dimensao: %s\n",             ____________________________________
          str1, sizeof( str1 ) );                   ____________________________________
  return 0;                                         ____________________________________
}                                                   ____________________________________
```

Apenas para as linhas que considera que estão erradas, apresente à direita uma versão dessa linha devidamente corrigida.

Q-14 [1.0 val.] Considere o seguinte programa em que as variáveis i1 e i2 estão guardadas no endereços 0x80000 e 0x80004, respectivamente.

```
int i1 = 1;
int i2 = 2;
int *pi1 = &i1;
int *pi2 = &i2;

*pi2 = 2;
pi1 = pi2;
*pi1 = 3;
```

Quais são os valores das variáveis no fim do programa? i1 = _____ i2 = _____ pi1 = _____ pi2 = _____

Q-15 Considere a seguinte função, escrita em C, que retorna o valor da dezena imediatamente a seguir ao valor dado como argumento. Por exemplo, se arg tiver o valor 12 retornará 20, se arg tiver o valor 80 retornará 80, se arg tiver o valor 156 retornará 160.

```
int proxima_dezena (int arg) {
  int pd = 0;

  return pd;
}
```

a) [1.0 val.] Implemente o código em falta na função, delimitado pelo rectângulo.

b) [1.5 val.] Complete o programa seguinte que lê um número do teclado e, recorrendo à função proxima_dezena, imprime o valor da próxima dezena.

```
_______ main() {
  int d, n;

  printf("introduza n: ");
  scanf(_____________________);

  ___ = proxima_dezena(____);
  printf("Proxima dezena de __ e' ___ \n", ____ , ____) ;

  _____________________________;
}
```

Q-16 Considere a função n_plica que permite multiplicar por n o valor de qualquer variável inteira.

a) [1.0 val.] Complete a implementação da função

```
void n_plica( _________ var, int n ) {
        _______ = n* _______;
}
```

b) [1.0 val.] Complete o exemplo da sua chamada para duplicar o valor na variável **int** a: n_plica(_____, ____);

c) [1.5 val.] Recorrendo a n_plica, implemente agora uma nova função que multiplica por n todos os elementos de um vector de inteiros de dimensão vec_size **Esta função tem de ser implementada com recurso a apontadores, ou seja, não pode utilizar a notação de vectores**.

```
void n_plicavec( int *vec, int vec_size, int n ) {

}
```