

**Resposta:**

RegDst tem que ser igual a 0 porque os bits que indicam o registrador a ser escrito (regB) são os bits 18 a 16.

EscReg tem que ser 1 porque deve haver escrita no registrador regB.

EscMem tem que ser 0 porque a instrução não escreve na memória de dados.

LerMem tem que ser 0 porque a instrução não lê da memória de dados.

UALfonte deve ser 1 porque o valor a ser adicionado ao conteúdo de regA deve ser o conteúdo do valor imediato.

DvC deve ser 0 porque a próxima instrução a ser executada deve ser aquela que se encontra no endereço PC+1.

UALParaReg deve ser 1 porque o valor a ser carregado no registrador a ser escrito regB deve ser a saída da UAL.

2. Considere o esquema de pipeline mostrado para a arquitetura vista em sala de aula, onde temos 5 estágios de pipeline: Busca (B), Decodificação/Leitura de regs (DL), Execução da instrução (EX), Acesso à memória de dados (MD), Escrita nos registradores (ER). Lembre-se que a escrita em um registrador é realizada na primeira metade do ciclo de relógio e a leitura de um registrador é realizada na segunda metade.

Analise o trecho de código abaixo e indique as instruções onde poderá existir conflito estrutural, ou de controle ou de dados, quando este código for executado. Suponha que a única maneira de resolver os conflitos é atrasar a execução dos estágios através da inserção de instruções noops. Indique o número de instruções noops que deverão ser inseridas para resolver cada conflito.

```
addi 3 3 1
lw 3 4 0
add 4 2 2
add 5 6 5
addi 7 7 1
addi 3 3 1
beq 2 5 if
addi 1 3 1
if addi 1 1 1
halt
```

**Resposta:**

```
addi 3 3 1 /* conflito de dados porque lw posterior precisa
do conteúdo do registrador 3 */
lw 3 4 0 /* conflito de dados porque add posterior
precisa do conteúdo do registrador 4 */
add 4 2 2
add 5 6 5 /* conflito de dados porque beq posterior
precisa do conteúdo do registrador 5 */
addi 3 3 1
beq 2 5 if /* conflito de controle */
addi 1 3 1
if addi 1 1 1
halt
```

Inserindo instruções noop para remover os conflitos:

```
addi 3 3 1 B DL EX MD ER - 2 inst. noop
lw 3 4 0 B DL EX MD ER - 2 inst. noop
add 4 2 2 B DL EX MD ER
add 5 6 5 B DL EX MD ER
addi 3 3 1 B DL EX MD ER - 1 inst. noop
beq 2 5 if B DL EX MD ER - 2 inst. noop
addi 1 3 1 B DL EX MD ER
addi 1 1 1 B DL EX MD ER
halt B DL EX MD ER
```

O código deve ficar com mostrado abaixo:

```
        addi 3 3 1
        noop
        noop
        lw 3 4 0
        noop
        noop
        add 4 2 2
        add 5 6 5
        addi 3 3 1
        noop
        beq 2 5
        noop
        noop
        addi 1 3 1
if      addi 1 1 1
        halt
```

3. Explique o que são e como funcionam os processos de compilação, montagem e ligação.

**Resposta:**

**Compilação:**

Processo que consiste na análise de um programa escrito em linguagem de alto nível (programa fonte) e sua tradução em um programa em linguagem de máquina. Este processo é realizado pelo Compilador. A Análise feita pelo compilador consiste em 3 partes:

- A análise léxica onde o programa fonte é decomposto em seus elementos individuais (comandos, operadores, variáveis, etc), gerando erros se for encontrada alguma incorreção.
- A análise sintática onde são criadas as estruturas de cada comando e verificação de acordo com as regras gramaticais da linguagem.
- A análise semântica onde são verificadas as regras semânticas estáticas, podendo produzir mensagens de erros.

Após esta análise, se traduz o programa para linguagem de montagem ou para linguagem de máquina. Caso o programa seja traduzido para linguagem de montagem, utiliza-se um montador para traduzi-lo para linguagem de máquina. Depois, realiza-se o processo de ligação deste código em linguagem de máquina para finalizar a resolução de todos os símbolos não resolvidos anteriormente.

**Montagem:**

Processo que consiste em traduzir um programa em linguagem de montagem (assembly) para seu equivalente escrito em linguagem de máquina. Este processo é realizado pelo montador. Esta tradução consiste em substituir os códigos de operação simbólicos por valores numéricos, nomes simbólicos de endereços por valores numéricos e converter valores de constantes para valores binários.

**Ligação:**

Processo onde é feita a identificação de chamadas a rotinas e respectiva conexão entre o código objeto principal e o código das rotinas. Este processo é executado pelo ligador. O ligador examina o código-objeto, procura referências externas não resolvidas e suas localizações nas bibliotecas substituindo a linha de chamada pelo código da rotina e emitindo mensagem de erro em caso de não encontrar a rotina.

4. Considere um sistema que possui um controlador de teclado com três registradores utilizados para se comunicar com o resto do sistema: um para receber comandos da UCP, outro para indicar se o controlador de teclado está pronto para enviar um byte para a UCP, e outro para armazenar o byte que corresponde a tecla inserida pelo usuário. O primeiro registrador possui o endereço 05, o segundo 06 e o terceiro 07. Para se receber um byte correspondente a uma tecla inserida pelo usuário, deve-se enviar o comando 01 para o controlador de teclado, utilizando o endereço 05. O controlador de teclado armazena o valor 01 no registrador de estado (registrador 06) quando uma tecla foi inserida pelo usuário e o byte correspondente a ela se encontra no registrador 07. O controlador de teclado pode trabalhar por interrupção e existe também um controlador de acesso direto à memória. Suponha que se deseja transferir 20 bytes do teclado para a memória. Descreva **detalhadamente** os três possíveis métodos que podem ser utilizados: por E/S programada, por interrupção e por acesso direto à memória.

**Resposta:**

E/S programada:

A UCP envia o comando 01 para o registrador com endereço 05. Depois, fica lendo o conteúdo do registrador com endereço 06 e verificando se o seu conteúdo é igual a 01. Quando o conteúdo for igual a 01, a UCP realiza a operação de leitura do registrador com endereço 07. Este procedimento é repetido até que todos os 20 bytes tenham sido transferidos do teclado.

E/S por interrupção:

A UCP envia o comando 01 para o registrador com endereço 05. Depois, vai executar outras instruções. Quando o controlador de teclado possui um byte armazenado referente a uma tecla, ele envia um sinal de interrupção para a UCP. A UCP, ao receber o sinal de interrupção, finaliza a instrução que estiver executando, salva o contexto do programa que estava sendo executado e atende a interrupção, lendo o byte que está armazenado no registrador de endereço 07. Este procedimento é repetido até que todos os 20 bytes tenham sido transferidos do teclado.

E/S por acesso direto à memória:

Neste caso, existe um controlador de DMA (Direct Access Memory) que é responsável pela transferência dos bytes. A UCP informa a este controlador que a operação de transferência deverá ser realizada com o controlador de teclado, o endereço inicial da memória onde os bytes do teclado devem ser armazenados, o número de bytes a serem recebidos e a indicação de que o controlador deve transferir bytes do controlador de teclado para a memória. Após receber estas informações, o controlador realiza a transferência dos bytes entre memória e controlador de teclado sem a intervenção da UCP. Após a transferência de todos os 20 bytes, o controlador de DMA avisa o fim da operação para a UCP através de um sinal de interrupção.

5. Considere os seguintes tipos de interface de E/S: por programa, por interrupção e por acesso direto à memória.
- a) Descreva, em termos gerais, a operação de cada uma delas.

**Resposta:**

E/S por programa: O processador tem controle direto sobre a operação de E/S, incluindo a detecção do estado do dispositivo, o envio de comandos de leitura ou escrita e transferência de dados. Para realizar uma transferência de dados, o processador envia um comando para o módulo de E/S e fica monitorando o módulo para identificar o momento em que a transferência pode ser realizada. Após detectar que o módulo está pronto, a transferência de dados é realizada através do envio de comandos de leitura ou escrita em registradores do controlador do dispositivo de E/S pelo processador. Se o processador for mais rápido que o módulo de E/S, o procedimento realizado para identificar quando o módulo está pronto representa um desperdício de tempo de processamento.

E/S por interrupção: Neste caso, o processador envia um comando para o módulo de E/S e continua a executar outras instruções, sendo interrompido pelo módulo quando ele estiver pronto para realizar a transferência de dados, que é executada pelo processador através da obtenção dos dados da memória principal, em uma operação de saída, e por armazenar dados na memória principal, em uma operação de entrada.

E/S por acesso direto à memória (DMA): Nesse caso a transferência de dados entre o módulo de E/S e a memória principal é feita diretamente sem envolver o processador. Existe um outro módulo denominado controlador de DMA que realiza a transferência direta de dados entre a memória e o módulo de E/S. Quando o processador deseja efetuar a transferência de um bloco de dados com um módulo de E/S, ele envia um comando para o controlador de DMA indicando o tipo de operação a ser realizada (leitura ou escrita de dados), endereço do módulo de E/S envolvido, endereço de memória para início da operação de leitura ou escrita de dados e número de palavras a serem lidas ou escritas. Depois de enviar estas informações ao controlador de DMA, o processador pode continuar executando outras instruções. O controlador de DMA executa a transferência de todo o bloco de dados e ao final envia um sinal de interrupção ao processador, indicando que a transferência foi realizada.

- b) Considere um sistema onde o número de ciclos de relógio para realizar uma operação por programa é igual a 400, o processador utiliza um relógio de 500 MHz para executar as

instruções e nenhuma transferência de dados pode ser perdida. Determine o overhead – relação entre ciclos de CPU consumidos em um segundo pelo processo de Entrada/Saída e número de ciclos disponíveis- que ocorre quando se utiliza a interface por programa para os seguintes dispositivos:

b.1) Um mouse que deve ser interrogado pelo sistema 30 vezes por segundo para garantir que nenhum movimento dele seja perdido.

**Resposta:**

Como o mouse deve ser interrogado 30 vezes por segundo e em cada atendimento são gastos 400 ciclos, teremos 12000 ciclos sendo gastos em um segundo para realizar operações de E/S por programa. A frequência do relógio desta máquina é 500 MHz, ou seja,  $500 \times 10^6$  ciclos de relógio são realizados em um segundo. O overhead é calculado como a razão do número de ciclos utilizados em um segundo pelo método de E/S por programa dividido pelo número total de ciclos que são fornecidos em um segundo  $12000/(500 \times 10^6)=0,0024 \%$ .

b.2) Um disco flexível que transfere dados para o processador em blocos de 16 bits e possui uma taxa de transferência de dados de 50 KB/segundo.

**Resposta:**

Como a taxa de transferência é igual 50 KB/s e a unidade de transferência é 16 bits, ou seja, 2 bytes, temos que a taxa de atendimento deve ser igual a  $50 \text{ KB}/2\text{B}=25 \text{ K}$  vezes por segundo.

Então serão gastos  $25\text{K} \times 400$  ciclos de relógio em um segundo para realizar operações de E/S por programa. O “overhead” é igual a  $25 \times 10^3 \times 400/(500 \times 10^6)=2\%$

b.3) Um disco rígido que transfere dados para o processador em blocos de 16 bytes e possui uma taxa de transferência de 4MB/segundo.

**Resposta:**

Como a taxa de transferência é igual 4 MB/s e a unidade de transferência é 16 bytes, temos que a taxa de atendimento deve ser igual a  $4 \text{ MB}/16\text{B}=250 \text{ K}$  vezes por segundo.

Então serão gastos  $250\text{K} \times 400$  ciclos de relógio em um segundo para realizar operações de E/S por programa. O “overhead” é igual a  $250 \times 10^3 \times 400/(500 \times 10^6)=20\%$

- c) Discuta a vantagem que a interface por interrupção possui em relação à interface por programa. Ilustre sua resposta calculando a fração de tempo de CPU consumida pelo disco rígido conforme descrito no item b.3 assumindo que o overhead de cada transferência, incluindo a interrupção, é 500 ciclos de relógio e que o disco está ativo em 5 % do tempo total em que a CPU está sendo utilizada. Porque a porcentagem do tempo que um dispositivo está ativo é importante para comparar as interfaces por programa e por interrupção?

**Resposta:**

Como a taxa de transferência do disco rígido é igual 4 MB/s e a unidade de transferência é 16 bytes, temos que a taxa de atendimento deve ser igual a  $4\text{MB}/16\text{B}=250 \text{ K}$  vezes por segundo.

Então serão gastos  $250\text{K} \times 500$  ciclos de relógio em um segundo para realizar operações de E/S por interrupção. Caso o dispositivo estivesse ativo todo o tempo, o “overhead” seria igual a  $250 \times 10^3 \times 500/(500 \times 10^6)=25\%$  que é maior que o “overhead” da E/S por programa. Mas como o dispositivo só está ativo 5% do tempo total, neste caso os ciclos de relógio só serão utilizados quando o dispositivo interromper o processador, ou seja, o “overhead” será  $25\% \times 5\% = 1,25\%$ .

A porcentagem do tempo que um dispositivo está ativo é importante para comparar estes dois métodos, porque no método por E/S por programa o processador é obrigado a gastar ciclos de relógio para monitorar o dispositivo independentemente do fato de ele estar sendo acionado ou não. No caso do método por interrupção, os ciclos só serão gastos quando o dispositivo estiver efetivamente sendo acionado para realizar transferência de dados.

Neste caso, observe que na operação por programa, de cada 100 tentativas em que o processador acessa o dispositivo apenas em 5 este estará disponível para transferir informações (5% ativo). Na operação de interrupção, o processador apenas será interrompido quando o dispositivo de E/S estiver em atividade.

- d) Considere agora outro cenário, onde o disco rígido é controlado por um controlador de DMA, cada transferência entre o disco e a memória ocorre em blocos de 8KB e ocorrem transferências em 100 % do tempo total que a CPU está sendo utilizada. Calcule a fração de tempo de CPU que é consumida, caso necessite-se, em cada transferência, de 1000 ciclos de relógio do processador para inicializar o controlador de DMA e o tratamento da interrupção gerada pela

finalização da operação do controlador do DMA necessite de 500 ciclos de relógio do processador. Ignore qualquer impacto que possa ocorrer devido à contenção do barramento entre o processador e o controlador de DMA.

**Resposta:**

O número de ciclos de inicialização é igual a 1000 e de interrupção é igual a 500. Logo o número total de ciclos por operação de DMA é igual a 1500. Como descrito no item b.3, o disco possui uma taxa de transferência de 4 MB/s. Neste caso, são transferidos 8 KB por operação de transferência de dados, então cada operação de DMA terá a duração de  $8\text{KB}/4\text{ MB/s} = 2 \times 10^{-3}$  segundos. Se as transferências ocorrem 100 % do tempo, teremos que o número de ciclos gasto para transferência de dados será  $1500 / (2 \times 10^{-3}) = 750 \times 10^3$  ciclos de clock/s . O “overhead” será  $(750 \times 10^3) / (500 \times 10^6) = 0,15 \%$