AD2 - Organização de Computadores 2013.1
Data de entrega:14/05/2013

1. (1,5) Compare máquinas de 1, 2 e 3 endereços escrevendo programas para calcular:

X=(A+BxC)/(D-ExF)

As instruções disponíveis para uso são as seguintes:

1 Endereço:	2 Endereços:	3 Endereços:
LOAD M	MOV (X=Y)	MOV (X=Y)
STORE M	ADD (X=X+Y)	ADD (X=Y+Z)
ADD M	SUB (X=X-Y)	SUB (X=Y-Z)
SUB M	MUL (X=X*Y)	MUL (X=Y*Z)
MUL M	DIV (X=X/Y)	DIV (X=Y/Z)
DIV M		

M é um endereço de memória de 16 bits, e , X, Y e Z são ou endereços de memória de 16 bits ou de registradores de 4 bits. A máquina de 1 endereço usa um acumulador, e as outras duas têm 16 registradores e instruções operando sobre todas as combinações de endereços de memória e registradores. SUB X,Y subtrai Y de X e SUB X,Y,Z subtrai Z de Y e coloca o resultado em X. Assumindo códigos de operação de 8 bits e comprimentos de instruções que são múltiplos de 4, quantos bits cada máquina precisa para calcular X?

Máquina 1: instruções com 1 endereço (ou 1 operando) X = (A+BxC)/(D-ExF)

```
LOAD B          =>    ACC   <- B
MUL C           =>    ACC   <- ACC * C
ADD A           =>    ACC   <- ACC + A
STORE X         =>    X     <- ACC
LOAD E          =>    ACC   <- E
MUL F           =>    ACC   <- ACC * F
STORE T1        =>    T1    <= ACC
LOAD D          =>    ACC   <= D
SUB T1          =>    ACC   <= ACC – T1
STORE T1        =>    T1    <= ACC
LOAD X          =>    ACC   <= X
DIV T1          =>    ACC   <= ACC / T1
STORE X         =>    X     <= ACC
```

Máquina 2: instruções com 2 endereços (ou operandos)

(a) Sem perda de conteúdo

```
MOV X,B     =>  X □ B
MUL X,C     =>  X □ X * C
ADD X,A     =>  X □ X + A
MOV T1,E    =>  T1 □ E
MUL T1,F    =>  T1 □ T1 * F
MOV T2,D    =>  T2 □ D
SUB T2,T1   =>  T2 □ T2 – T1
DIV X,T2    =>  X □ X / T2
```

(b) Com perda de conteúdo

```
MUL B,C     =>  B □ B * C
```

```
ADD A,B     =>  A ← A + B
MUL E,F     =>  E ← E * F
SUB D,E     =>  D ← D - E
DIV A,D     =>  A ← A / D
```

Máquina 3: instruções com 3 endereços (ou 3 operandos):

```
MUL  X, B, C      =>    X = B * C
ADD  X, A, X      =>    X = A + X
MUL T1, E, F      =>    T1 = E * F
SUB T1, D, T1     =>    T1 = D - T1
DIV  X, X, T1     =>    X = X / T1
```

Máquina 1: *Cada instrução é composta de 1 código de operação (4 bits) e um operando que corresponde a um endereço de memória (16 bits), logo, cada instrução terá 20 bits de tamanho. Ao todo foram 13 instruções, então, o total de bits para o programa é : 13 instruções x 20bits por instrução =* ***260 bits***

Máquina 2: Cada instrução é composta de 1 código de operação (4 bits) e dois operandos onde cada um corresponde a um endereço de registrador (4 bits para endereçar 16 registradores), logo, cada instrução terá 12 bits de tamanho. Ao todo foram 8 instruções em (a) e 5 em (b), então, o total de bits para o programa é : em (a) 8 instruções x 12bits por instrução = ***96 bits*** *e em (b) 5 instruções x 12bits por instrução =* ***60 bits***

Máquina 3: Cada instrução é composta de 1 código de operação (4 bits) e três operandos que cada um corresponde a um endereço de registrador (4 bits para endereçar 16 registradores), logo, cada instrução terá 16 bits de tamanho. Ao todo foram 5 instruções, então, o total de bits para o programa é : 5 instruções x 16 bits por instrução = ***80 bits***

2. (1,0) Explique o que são e como funcionam os processos de montagem, compilação e ligação.

Montagem : *Processo que consiste em traduzir um programa em linguagem de montagem (assembly) para seu equivalente em binário. Processo este realizado pelo montador. Esta tradução consiste em substituir a partir dos programas os códigos de operação simbólicos por valores numéricos, nomes simbólicos de endereços por valores numéricos e converter valores de constantes para valores binários. Tipicamente, em montadores de dois passos, o programa é examinado instrução por instrução duas vezes. No primeiro passo são detectados os erros e montada a tabela de símbolos de endereços. No segundo passo é feita a criação do código objeto.*

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 (programa objeto). Processo este 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.

Ligação: *Processo onde é feita a interpretação à chamada a uma rotina e respectiva conexão entre o código objeto principal e o código da rotina. 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 emitindo mensagem de erro em caso de não encontrar a rotina.*

3. (1,0) Descreva as categorias da classificação de arquiteturas segundo Flynn.

Consiste em uma das formas mais comuns de classificação de processamento paralelo. São estas as categorias de sistemas de computação:

SISD *- Single instruction stream, single data stream. Um único processador executa uma única seqüência de instruções sobre dados armazenados em uma única memória. Exemplo: Processadores de computadores pessoais.*

SIMD *– Single instruction stream, multiple data stream. Vários elementos de processamento. Cada um tem uma memória de dados. Cada instrução é executada sobre um conjunto de dados diferente. Exemplo: Processadores matriciais.*

MISD *– Multiple instruction stream, single data stream. A seqüência de dados é transmitida para um conjunto de processadores, cada um dos quais executa uma seqüência de instruções diferente. Não existem processadores comerciais que utilizam este modelo.*

MIMD *– Multiple instruction stream, multiple data stream. Conjunto de processadores executa simultaneamente seqüências diferentes de instruções sobre conjuntos de dados diferentes. Exemplo: SMPs, clusters, sistemas NUMA.*

4. (1,5) Responda:

a) Dados os valores de memória abaixo e uma máquina de 1 endereço com um acumulador:

palavra 20 contém 40
palavra 30 contém 50
palavra 40 contém 60
palavra 50 contém 70

Quais valores as seguintes instruções carregam no acumulador?

-Load imediato 20
-Load direto 20
-Load indireto 20

- LOAD IMEDIATO 20
Resposta: Nesta instrução o valor a ser colocado no acumulador corresponde ao valor fornecido como operador, portanto:
ACC <- 20 (o valor a ser colocado no acumulador é 20)

- LOAD DIRETO 20
Resposta: Nesta instrução o valor a ser colocado no acumulador corresponde ao valor contido no endereço de memória fornecido como operador, portanto:
ACC <- (20) (o valor a ser colocado no acumulador é 40)

- LOAD INDIRETO 20
Resposta: Nesta instrução o valor a ser colocado no acumulador corresponde ao valor contido no endereço que consta como valor no endereço de memória fornecido como operador, portanto:
ACC <- ((20)) (o valor a ser colocado no acumulador é 60)

b) Analise os modos de endereçamento direto, indireto e imediato, estabelecendo diferenças de desempenho, vantagens e desvantagens de cada um.

Imediato: O campo operando contém o dado / Vantagem: Rapidez na execução da instrução / Desvantagem: Limitação do tamanho do dado inadequado para o uso com dados de valor variável / Não requer acesso à memória principal. Mais rápido que o modo direto

Direto: O campo operando contém o enderço do dado / Vantagem: Flexibilidade no acesso a variáveis de valor diferente em cada execução do programa / Desvantagem: Perda de tempo, se o dado é uma constante / Requer apenas um acesso à memória principal. Mais rápido que o modo indireto

Idireto: O campo de operando contém o endereço do dado / Vantagem: Manuseio de vetores (quando o modo indexado não está disponível). Usar como "ponteiro" / Desvantagem: Muitos acesso à MP para execução / Requer 2 acessos à memória principal.

c) Qual é o objetivo do emprego do modo de endereçamento base mais deslocamento? Qual é a diferença de implementação entre esse modo e o modo indexado?

O base mais deslocamento tem como seu principal objetivo permitir a modificação de endereço de programas ou módulos destes (que é a realocação de programa), bastando para isso uma única alteração no registrador base. O base mais deslocamento tem como característica o endereço a ser obtido da soma do deslocamento com o registrador base. A diferença entre os modos é mais conceitual do que na implementação. O base mais deslocamento difere do modo indexado onde o do registrador base é fixo e ocorre variação no deslocamento, pois ao contrário, em endereçamento base mais deslocamento o deslocamento é fixo e há alteração do registrador base.

5. (2,0) Considere um computador, cuja representação para ponto fixo e para ponto flutuante utilize 20 bits. Na representação em ponto flutuante, as combinações possíveis de bits representam números normalizados do tipo $+/-(1,b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{7}b_{8}b_{-9}b_{-10}b_{-11}b_{-12}\ b_{-13} \times 2^{\text{Expoente}})$, onde o bit mais à esquerda representa o sinal (0 para números positivos e 1 para números negativos), os próximos 6 bits representam o expoente em complemento a 2 e os 13 bits seguintes representam os bits b_{-1} a b_{-13}, como mostrado na figura a seguir:

S	Expoente representado em complemento a 2	$b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}\ b_{-5}\ b_{-6}\ b_{-7}\ b_{-8}\ b_{-9}\ b_{-10}\ b_{-11}\ b_{-12}\ b_{-13}$
1	6	13

a) (0,4) Considere o seguinte conjunto de bits representado em hexadecimal B00A0. Indique o valor deste número **em decimal**, considerando-se que o conjunto representa:

$(B00A0)_{16} = (1011\ 00000000\ 10100000)_2$

(a.1) um inteiro sem sinal

$2^{19} + 2^{17} + 2^{16} + 2^7 + 2^5 = 721.056$

(a.2) um inteiro em sinal magnitude

$-(2^{17} + 2^{16} + 2^7 + 2^5) = -196.768$

(a.3) um inteiro em complemento a 2

$-2^{19} + (2^{17} + 2^{16} + 2^7 + 2^5) = -327.520$

(a.4) um real em ponto flutuante conforme descrição do enunciado.

$(1011\ 00000000\ 10100000)_2$

Sinal: 1 (negativo)

Expoente: 011000 (complemento a 2) = $+(2^4 + 2^3) = +24$

Mantissa: ,0000010100000

Resposta: -1,00000101 x 2^{-+24}

$(-10000010100000000000000000)_2$ = -17.104.896

b) (0,2) Qual será a representação em ponto flutuante dos seguintes valores decimais neste computador:

b.1) +1093,72

Convertendo para binário = 10001000101,1011100001010001 =>
normalizando 1,00010001011011100001010001 x 2^{10}
Temos então:
Sinal = 0 (positivo)
Expoente (complemento a 2) = +10 = 001010
Mantissa = 00010001011011100001010001
Resultado: 0 001010 0001000101101

b.2) –7,7

Convertendo para binário = 111,10110011001100110011001...
=> normalizando 1,11101100110011001... x 2^2
Temos então:
Sinal = 1 (negativo)
Expoente (complemento a 2) = +2 = 000010
Mantissa = 111011001100110011..
Resultado: 1 000010 1110110011001.

c) (0,6) Indique o menor e o maior valor positivo normalizado na representação em ponto flutuante para este computador. Mostre os valores **em decimal**.

Menor positivo 0 100000 00000000000000 = 1,00000000000 x 2^{-32} = +0,00000000023283064
Maior positivo 0 011111 11111111111111 = 1,11111111111 x 2^{+31} = 4.294. 967.296

d) (0,8) Caso se deseje utilizar a representação em excesso para representar o expoente, indique o excesso a ser utilizado e o menor e o maior valor positivo normalizado para esta nova representação. Mostre os valores **em decimal**.

Excesso = 2^{e-1} – 1, sendo e = 6, excesso=31

Obs.: Situações quando o expoente é todo 0 ou todo 1 são consideradas casos especiais. Estes casos especiais servem para representação do 0 ou não número. Por não mencionar os casos especiais no títuto da questão, não foram considerados na resposta.

Menor positivo 0 000000 00000000000000 = 1,00000000000 x $2^{-31=}$ +0,00000000046566129
Maior positivo 0 111111 11111111111111 = 1,11111111111 x 2^{+32} = 8.589.934.592

6. (1,0) Explique como funcionam os três mecanismos utilizados para transferir dados entre um dispositivo de E/S e a memória de um sistema de computação: por programa (polling), por interrupção e por acesso direto à memória.

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 pelo processador. Se o processador for mais rápido que o módulo de E/S., essa espera representa um desperdício de tempo de processamento. As vantagens deste método são: hardware simples e todos os procedimentos estão sobre controle da UCP. As desvantagens são: utilização do processador para interrogar as interfaces, o que acarreta perda de ciclos de processador que poderiam ser utilizados na execução de outras instruções, : utilização do processador para realizar a transferência de dados, o que também acarreta perda de ciclos de processador.*

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. A vantagem deste método é que não ocorre perda de ciclos de processador para interrogar a interface, já que neste caso, não se precisa mais interrogar a interface, ela avisa quando pronta. As desvantagens são: necessidade de um hardware adicional (controlador de interrupções , por exemplo), gerenciamento de múltiplas interrupções e perda de ciclos de relógio para salvar e recuperar o contexto dos programas que são interrompidos.*

E/S por 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. As vantagens deste método são: permite transferência rápida entre interface e memória porque existe um controlador dedicado a realizá-la e libera a UCP para executar outras instruções não relacioandas a entrada e saída. A desvantagem é que precisamos de hardware adiciona*

7. (1,0) Explique como funciona um barramento SCSI (sugestões de fonte de consulta: livro do Stallings e a página http://informatica.hsw.com.br/scsi.htm)

O barramento SCSI (Small Computer System Interface) é um barramento de comunicação rápida que conecta vários dispositivos a um computador ao mesmo tempo, incluindo discos rígidos, scanners, CD-ROM/RW e impressoras. O SCSI tem vários benefícios: é muito rápido - chega a 320 Mb/s - e está no mercado há mais de 20 anos, sendo incansavelmente testado - portanto, tem a reputação de ser confiável. Da mesma maneira que o Serial-ATA eo Fire-Wire, ele aceita a ligação de vários dispositivos diferentes em um barramento.
No entanto, o SCSI também tem alguns problemas. Tem suporte limitado a BIOS do sistema e precisa ser configurado para cada computador. Também não existe uma interface de software comum para SCSI. Por fim, todos os tipos diferentes de SCSI têm velocidades, largura de barramento e conectores diferentes, o que pode ser confuso. Mas, uma vez conhecido o significado de "Fast", "Ultra" e "Wide", fica fácil entender.

O SCSI tem três especificações básicas:

1. *SCSI-1: especificação original desenvolvida em 1986, está obsoleta. Tinha largura de barramento de 8 bits e velocidade de clock de 5 MHz.*
2. *SCSI-2: adotada em 1994, esta especificação incorporou o Common Command Set - CCS (Conjunto de Comandos Comuns) - 18 comandos considerados imprescindíveis para o suporte a qualquer dispositivo SCSI. Também tinha a possibilidade de dobrar a velocidade de clock para 10 MHz (Fast), dobrar a largura de barramento para 16 bits e aumentar o número de dispositivos para 15 (Wide), ou ambos (Fast/Wide). O SCSI-2 também incorporou o enfileiramento de comandos, permitindo que os dispositivos armazenassem e priorizassem comandos do computador onde estavam instalados.*
3. *SCSI-3: especificação de 1995 que trouxe uma série de padrões menores dentro de sua abrangência geral. Um conjunto de padrões envolvendo a Interface Paralela SCSI (SPI), que é a forma como os dispositivos SCSI se comunicam, continuou a evoluir dentro do SCSI-3. A maioria das especificações SCSI-3 começam com o termo Ultra: Ultra para variações SPI, Ultra2 para variações SPI-2 e Ultra3 para variações SPI-3. As designações Fast (rápido) e Wide (largo) funcionam como no SCSI-2. SCSI-3 é o padrão atualmente em uso.*

Combinações diferentes de velocidade de barramento dobrada e de clock dobrada, e especificações SCSI-3 resultaram em muitas variações SCSI. Além da velocidade de barramento expandida, o Ultra320 SCSI usa transferência de dados em pacotes, aumentando sua eficiência. O Ultra2 também foi o último modelo a ter largura de banda "estreita", de 8 bits.
Os tipos citados até então de SCSI são paralelos: bits de dados movem-se simultaneamente pelo barramento, ao invés de um por vez. O tipo mais novo de SCSI, chamado de Serial Attached SCSI (SAS), usa comandos SCSI, mas transmite dados de forma serial. O SAS usa uma conexão ponto-a-ponto para mover dados a 3 Gb/s, e cada porta SAS suporta até 128 dispositivos ou expansores.
Os tipos diferentes de SCSI usam controladoras e cabos para se comunicar com os dispositivos.

8. (1,0) Verifique se é válida a seguinte definição para overflow (estouro) em uma operação aritmética na representação complemento a dois utilizando-se *n* bits: Caso os bits "vai-um" nas posições *n-1* e *n* sejam diferentes ocorreu overflow. Caso contrário não ocorreu overflow.

Resposta: Esta definição é válida, como mostrado a seguir.
Considere dois números A e B representados utilizando-se n bits:

$A= (a_{n-1}a_{n-2}\ldots a_0)$
$B= (b_{n-1}b_{n-2}\ldots b_0)$

Os bits c_n e c_{n-1} correspondem aos bits "vai-um" nas posições *n* e *n-1* respectivamente

c_n c_{n-1}
a_{n-1} ..... a_0
+
b_{n-1} ..... b_0

Verificando cada caso possível dos valores dos bits a_{n-1}, $b_{n-1,}$ e c_{n-1}:

a_{n-1}	b_{n-1}	c_{n-1}	Overflow?	$c_n \# c_{n-1?}$
0	0	0	Não. Dois números positivos somados tem como resultado um número positivo	Não
0	0	1	Sim. Dois números positivos somados tem como resultado um número negativo	Sim
0	1	0	Não. Nunca ocorre overflow quando um número positivo é somado com um número negativo	Não
0	1	1	Não. Nunca ocorre overflow quando um número positivo é somado com um número negativo	Não

1	0	0	Não. Nunca ocorre overflow quando um número positivo é somado com um número negativo	Não
1	0	1	Não. Nunca ocorre overflow quando um número positivo é somado com um número negativo	Não
1	1	0	Sim. Dois números negativos somados tem como resultado um número positivo	Sim
1	1	1	Não. Dois números negativos somados tem como resultado um número negativo	Não