AD2 - Organização de Computadores 2007.2
Gabarito

"Atenção: Como a avaliação a distância é individual, caso seja constatado que provas de alunos distintos são cópias umas das outras, independentemente de qualquer motivo, a todas será atribuída a nota ZERO. As soluções para as questões podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual."

1. (1,25) Descreva em detalhes os tipos de operações do Pentium II e do Power PC. (pesquise no livro: Arquitetura e Organização de Computadores, de William Stallings)

Os tipos de operações do Pentium II são:

- ⇒ **Transferência de dados:**
  Consiste em instruções com a função de transportar dados especificados em um operando, este que pode corresponder a um endereço de memória, um registrador, topo de pilha ou o próprio valor para outro operando que pode ser da mesma forma um endereço de memória, registrador ou topo de pilha. A instrução que executa a transferência de dados deverá especificar os 2 operandos (fonte e destino), o tamanho dos dados e também deverá especificar o modo de endereçamento de cada operando.
  São exemplos: MOV (cópia entre operandos), PUSH (coloca o operando no topo da pilha), PUSHA (coloca todos os registradores na pilha), MOVSX (Cópia entre operandos com representação de complemento a dois), entre outros
- ⇒ **Instruções aritméticas:**
  São instruções que detém a função da execução das operações básicas aritméticas (como soma, subtração, multiplicação e divisão). São oferecidas tanto para números inteiros como para números com ponto flutuante. Além destas operações básicas, outras funções com apenas um operando poderão ser executadas como: obter um valor absoluto, negar o controle, incrementar o operando em 1, decrementar em 1. As instruções aritméticas podem envolver transferência de dados
  São exemplos: ADD (soma de operandos), SUB (subtração de operandos), MUL (multiplicação entre operandos), IDIV (divisão de números inteiros com sinal)
- ⇒ **Instruções lógicas:**
  Consistem em instruções que efetuam operações lógicas básicas entre unidades endereçáveis ou manipular bits individuais de uma palavra. As operações lógicas podem ser aplicadas bit a bit a unidades de dados lógicos de 0 bits, e ainda operações de manipulação de bits como deslocamento aritmético e deslocamento cíclico, além de poder envolver transferência de dados
  São exemplos: AND (operação de E entre operandos), BTS (teste e altera valor de um bit, SHL/SHR (deslocamento lógico para esquerda / para direita), ROL/ROR (Rotação para a esquerda e para direita), entre outros.
- ⇒ **Instruções de transferência de controle:**
  Consistem em instruções que quebram a seqüência de execução das instruções onde a próxima instrução seria aquela que seque imediatamente na memória a instrução corrente.
  As instruções podem ser de desvio ou de salto:
  As instruções de desvio contêm como operando o endereço da próxima instrução a ser executada. Com freqüência, essa instrução é um desvio incondicional, desvio este que será feito (o registrador PC é atualizado com o endereço especificado no operando) apenas se a condição for satisfeita. Caso contrário, será executada a próxima instrução da seqüência.

As instruções de salto: são instruções onde o salto indica que a execução de uma instrução da seqüência de instruções deve ser omitida, o endereço da próxima instrução a ser executada será obtido somando o endereço da instrução corrente com o tamanho de uma instrução.
Além destas, outras instruções de transferência de controle poderão ser feitas como a chamada de procedimento que consiste em "invocar" ou mesmo "chamar" o procedimento, que consiste em instruir o processador a executar todo o procedimento e, então, retornar ao ponto em que ocorreu a chamada.
São exemplos: IMP (desvio incondicional), CALL (chamada de um procedimento), JE/JZ (Desvia se igual / Desvia se zero), LOOPE / LOOPZ (repete se igual / repete se zero), entre outros.

⇒ **Operações sobre seqüência de caracteres**
São exemplos: MOVS (copia byte, word, dword, sobre um elemento de uma sequência indexada pelos registradores ESI e EDI), LODS (carrega word, dword, qword de uma seqüência)

⇒ **Suporte para linguagem de alto nível**
São exemplos: ENTER (cria um registro de ativação na pilha, que pode ser representada para implementar regras de uma linguagem de alto nível estruturada em blocos), LEAVE (reverte a ação de uma instrução ENTER anterior), BOUND (verificação de limites de vetor)

⇒ **Operações sobre bits de condição:**
São exemplos: STC (ativa o CF ('vai um') do registrador de flags, LAHF (copia os bits SF, ZF, AF, PF, CF do registrador AH).

⇒ **Operações sobre registradores de segmento:**
São exemplos: LDS (carrega um apontador de segmento no registrador de segmento DS).

⇒ **Operações de controle de sistema**
Consistem em instruções que apenas podem ser executadas quando o processador está no estado privilegiado ou está executando um programa carregado em uma área especial de memória que é privilegiada, instruções estas reservadas para o uso pelo sistema operacional.
HLT (pára a execução do programa, suspendendo o processador), LOCK(bloqueia uma área a memória compartilhada, para que o Pentium II tenha uso exclusivo desta área durante a instrução seguinte ao LOCK), ESC (escape para uma extensão do processador), WAIT (espera até que o sinal de BUSY seja negado).

⇒ **Operações de proteção**
São exemplos: SGDT (armazena a tabela de descritor global), LSL (carrega o limite do segmento), YERR/YERW (verifica o segmento para leitura/escrita.

⇒ **Operações de gerenciamento da memória cache:** Corresponde a instruções com funções de gerenciamento da memória cache.
São exemplos: INVD (invalida a memória cache interna), WBINVD (invalida a memória cache interna, depois de gravar na memória as linhas),

Os tipos de operações do Power PC são:

⇒ **Instruções de desvio:** São oferecidas instruções de desvio condicional e incondicional usuais. As de desvio condicional pode testar um único bit do registrador de condição, desviando se ele tem o valor 0, ou se ele tem o valor 1, ou desviando independente do valor deste bit. Poderá desviar em função do valor do registrador contador, sendo seu valor 0 ou diferente de 0. O contador poderá ser decrementado de 1 antes do teste, sendo conveniente para laços de repetição. Os desvios incondicionais não dependem de um teste (ou condição) para o desvio, este será feito tomando como base o endereço no operando.
São exemplos: b (desvio incondicional), bl (desvia para endereço-alvo e coloca o endereço efetivo da instrução seguinte ao desvio no Registrador de Ligação), bc (desvio condicional baseado no conteúdo do registrador contador), sc (chamada ao sistema), trap (compara dois operandos e chama a rotina de tratamento de interrupção.

⇒ **Instruções de carga e armazenamento:** Diferente das instruções de lógico e aritméticas, onde as operações só podem ser feitas com registradores, as instruções de carga e armazenamento referenciam posições de memória,
São exemplos: lwzu (carrega palavra e estende com zeros para a esquerda), td (carrega palavara dupla), lmw (carrega múltiplas palavras), lswx carrega seqüência de bytes nos registradores),

⇒ **Instruções de aritmética inteira:** Correspondem às instruções de execução das operações básicas aritméticas (como soma, subtração, multiplicação e divisão).
São exemplos: add (soma os conteúdos de dois registradores e coloca o resultado no terceiro registrador), sub (idem ao add sendo a operação de subtração), multw (idem ao add para multiplicação), divd (idem ao add para divisão, sendo colocado o quociente no terceiro registrador).

⇒ **Operações lógicas e deslocamento:** Correspondem às instruções que efetuam operações lógicas básicas entre operandos, deslocamentos e rotações nos bits individuais de uma palavra.
São exemplos cmp (compara dois operandos e sinaliza 4 bits de condição no campo especificado do registrador de condição), crand (efetua a operação lógica AND de dois bits do registrador de condição), and (operação lógica and dos conteúdos de 2 registradores), cntlzd (conta números de bits consecutivos iguais a 0), rldic (efetua a rotação para esquerda de um registrador), sld (desloca os bits do registrador fonte para a esquerda e coloca o resuldado no registrador destino).

⇒ **Instruções de ponto flutuante** Operações de carga, aritméticas e armazenamento com ponto flutuante.
São exemplos: lfs (carrega o número de ponto flutuante de 32 bits da memória, armazenando no registrador), fadd (adiciona 2 registradores colocando o resultado no terceiro registrador), fmadd (multiplicação de 2 registradores, adicionando o resultado em um terceiro), fcmpu (compara 2 operandos de ponto flutuante e atualiza os bits de condição).

⇒ **Instruções de gerenciamento de cache** Corresponde a instruções com funções de gerenciamento da memória cache.
São exemplos: dcbf (pesquisa a memória cache de dados por um determinado endereço e executa a operação de esvaziamento), icbi (invalida o bloco da memória cache de instruções.).

2. (1,25) Descreva em detalhes os modos de endereçamento do Pentium II e do Power PC. (pesquise no livro: Arquitetura e Organização de Computadores, de William Stallings)

Os modos de endereçamento do Pentium II são:

⇒ **Modo imediato,** onde o operando é incluído na instrução, isto é, representa um valor a ser utilizado diretamente na instrução, seu tamanho poderá ser byte (8 bits), word (16 bits), Dword(32bits).
Ex. Operando = A (sendo A = conteúdo de um campo de endereço da instrução)

⇒ **Modo de operando registrador,** o valor a ser utilizado na instrução é localizado em um registrador. Para instruções de caráter geral, tais como transferências de dados e instruções aritméticas ou lógicas, o operando pode ser um dos registradores de propósito geral de 32bits, 16 bits ou 8 bits.
Ex. LA = R (sendo R = registrador)

⇒ **Modo de endereçamento por deslocamento:** O endereço relativo do operando é especificado como parte da instrução, como um deslocamento de 8, 16 ou 32 bits. O modo de endereçamento por deslocamento é usado em poucas máquinas porque implica em instruções com tamanho grande.
Ex. LA = (SR) + A (sendo SR = registrador de segmento, A = conteúdo de um campo de endereço na instrução)

⇒ **Modo base com deslocamento:** A instrução inclui um deslocamento a ser adicionado a um registrador-base, que pode ser qualquer registrador de uso geral. Possui utilização em compiladores para apontar para início da área de variáveis locais, indexamento de vetores, endereçamento de campos de registro, sendo o tamanho do campo o deslocamento e registrador-base o início do campo (ou registro).
Ex. LA = (SR) + (B) + A (sendo SR = registrador de segmento, B = registrador-base, A = conteúdo de um campo de endereço na instrução)

⇒ **Modo índice com fator de escala e deslocamento:** a instrução inclui um deslocamento que é somado a um registrador índice. O registrador índice pode ser qualquer um registradores de uso geral, exceto o ESP (utilizado para endereçamento de pilha). Para calcular o endereço efetivo, o conteúdo do registrador índice é multiplicado por um fator de escala igual a 1,2,4 ou 8, e então é adicionado ao deslocamento, sendo muito útil para vetores.
Ex. LA = (SR) + (I) x S + A (sendo SR = registrador de segmento, I = registrador índice, S = fator de escala, A = conteúdo de um campo de endereço na instrução)

⇒ **Modo base mais índice e deslocamento:** o endereço efetivo é obtido somando os conteúdos do registrador-base, do registrador índice e do deslocamento, tem sua uso direcionado para endereçamento de elementos em um vetor local e um registro de ativação localizado em pilha.
Ex. LA = (SR) + (B) + (I) + A (sendo SR = registrador de segmento, I = registrador índice, B = registrador-base, A = conteúdo de um campo de endereço na instrução)

⇒ **Modo base mais índice com fator de escala e deslocamento**: o conteúdo do registrador índice é multiplicado por um fator de escala e somado com o conteúdo do registrador-base e o deslocamento. Tem seu uso direcionado a vetores que estejam armazenados em um registro de ativação; sendo o tamanho dos elementos do vetor iguais a 2,

4 ou 8 bytes. Permite também indexação para matrizes de 2 dimensões, com elementos do mesmo tamanho do vetor citado.
Ex. LA = (SR) + (I) x S + (B) + A (sendo SR = registrador de segmento, I = registrador índice, S = fator de escala, B = registrador-base, A = conteúdo de um campo de endereço na instrução).

⇒ **Modo relativo:** Onde um deslocamento é adicionado ao valor do contador de programa, que aponta para a próxima instrução a ser executada, sendo o deslocamento tratado como um valor com sinal permitindo aumentar ou diminuir o valor do endereço no contador de programa.
Ex. LA (PC) + A (sendo PC = contador de instruções, A = conteúdo de um campo de endereço na instrução).

Os modos de endereçamento do PowerPC são:

Em função da tecnologia RISC, com poucos modos de endereçamento, o PowerPC classifica seus modos de endereçamento de acordo com o tipo de instrução:

⇒ **Carga e armazenamento:** correspondem a 2 formas de modos de endereçamento alternativos:
Endereçamento indireto onde a instrução inclui um deslocamento de 16 bits a ser adicionado a um registrador-base, que pode ser um dos registradores de uso geral. E ainda, a instrução poderá que endereço efetivo deve ser armazenado no registrador-base permitindo assim a indexação progressiva facilitando o uso em laços de repetição.
Ex. EA = (BR) + D => (EA = endereço efetivo, BR = registrador-base, D = deslocamento)
Endereçamento indexado indireto: onde a instrução especifica um registrador-base e um registrador índice, ambos podendo ser qualquer registrador de uso geral. Neste modo de enderaçemento também permite a opção de atualizar o conteúdo do registrador-base como endereço efetivo calculado.
Ex. EA = (BR) + (IR) => (EA = endereço efetivo, BR = registrador-base, IR = (registrador-índice)

⇒ **Instruções de desvio: correspondem a 3 modos de endereçamento:**
Endereçamento absoludo: onde o endereço efetivo da próxima instrução é obtido a partir de um valor imediato de 24 bits contido na instrução.
Ex. EA = I => (EA = endereço efetivo, I = valor imediato)
Endereçamento relativo: onde o valor imediato de 24bits (para instruções de desvio incondicional) ou o valor imediato de 14 bits (para instruções de desvio condicional) é também entendido como antes. O valor resultante é então adicionado ao contador de programa, sendo portanto um endereço relativo ao endereço de instrução corrente.
Ex. EA = (PC) + I => (EA = endereço efetivo, PC = contador de programa, I = valor imediato)
Endereçamento indireto: onde o endereço da próxima instrução é obtido no registrador de ligação ou no registrador contador.
Ex. EA =(L/CR) => (EA = endereço efetivo, L/CR = registrador de ligação)

⇒ **Instruções de aritméticas:** Onde todos os operandos devem estar contidos em registradores ou ser especificados como parte da instrução.
Endereçamento de registrador: O operando fonte ou destino pode ser qualquer um dos registradores de uso geral.
Ex. EA = GPR => (EA = endereço efetivo, GPR = registrador de uso geral)
Endereçamento imediato: o operando fonte é um valor de 16 bits, com sinal contido na instrução.
Ex. Operando = I => (I = valor imediato)

⇨ **Instruções aritméticas de ponto flutuante:** todos os operandos devem estar em registradores de ponto flutuante, isto é, apenas o modo de endereçamento de registrador pode ser usado.
Ex EA = EPR => (EA = endereço efetivo, FPR = registrador de ponto flutuante)

3. (2,0) Elabore dois programas para o cálculo da seguinte equação para os processadores Pentium II e Power PC, respectivamente:

X= A+(B*(C-A)+(D-E/B)*D)

OBS: Considerando as variáveis acima (A,B,C...) como endereços de memória, e instruções que operem com números inteiros, pois para cálculos com ponto flutuante as instruções seriam diferentes.

Instruções escolhidas para Pentium II (padrão de instruções IA-32):
Fonte de consulta (site www.intel.com):
Volume1; http://www.intel.com/design/processor/manuals/253666.pdf
Volume2: http://www.intel.com/design/processor/manuals/253667.pdf

Add dest, src (um exemplo: ADD *r32, r/m32) – arquivo consultado: volume1 / pág. 3-30).*
Soma o operando destino (primeiro operando) e o operando fonte (segundo operando) e então armazena o resultado no operando destino. O operando destino pode ser um registro ou um endereço de memória. O segundo operando pode ser um valor imediato, um registrador ou mesmo um endereço de memória. Só não poderão, ambos operadores, serem endereços de memória simultaneamente. Se for usar o acumulador como operando deverá fazê-lo como sendo o primeiro operando.

Expressão equivalente: DEST ← DEST + SRC;

Div oper (um exemplo: DIV *r/m32 - fonte de consulta: volume1 / pág. 3-264).*
Divide o conteúdo no registrador AX ou EAX (acumulador de 16 ou 32 bits respectivamente) pelo divisor que corresponde ao operando fornecido e coloca o quociente no registrador AX ou EAX e o resto no registrador DX ou EDX (16 ou 32 bits respectivamente). O operando poderá ser um registrador ou endereço de memória.

Mov dest, src (um exemplo: MOV *r/m32, imm32 – fonte de consulta: volume1 / pág. 3-600)*
Copia o segundo operando (operando fonte) para o primeiro operando (operando destino). O operando fonte pode ser um valor imediato, registrador de uso geral, registrador de segmento ou locação de memória. O operando destino pode ser um registrador de uso geral, registrador de segmento ou locação de memória.

Mul oper (um exemplo: MUL *r/m32 – fonte de consulta: volume1 / pág.3-695)*
Multiplica o conteúdo do registrador AX ou EAX (acumulador de 16 ou 32 bits respectivamente) pelo valor do segundo operando armazenando o resultado no registrador AX ou EAX. O operando poderá ser um registrador de uso geral ou endereço de memória.

Sub dest, src (um exemplo: Sub *r32, r/m32) – fonte de consulta: volume2 / pág. 4-350).*
Subtrai o segundo operando(operando fonte) do primeiro operando(operando destino) e armazena o resultado no operando destino. O operando destino pode ser um registro ou um endereço de memória; o operando fonte pode ser um imediato, registrador ou endereço de memória. Só não poderão, ambos operadores, serem endereços de memória simultaneamente. Se for usar o acumulador como operando deverá fazê-lo como sendo o primeiro operando.

X= A+(B*(C-A)+(D-E/B)*D)

Mov ECX, A	=> Move o conteúdo do endereço A para um registrador de uso geral (ex. ECX)
Sub C, ECX	=> Subtrai o conteúdo do endereço de memória C pelo registrador ECX armazenando o resultado em C
Mov EAX, B	=> Copia o conteúdo do endereço de memória B para o acumulador
Mul C	=> Multiplica o conteúdo do acumulador pelo conteúdo do endereço C
Mov X, EAX	=> Copia o conteúdo do acumulador para um endereço de memória (X por exemplo)
Mov EAX, E	=> Move o conteúdo do endereço E para o registrador acumulador (EAX)
Div B	=> Divide o conteúdo do acumulador (EAX) pelo conteúdo do endereço B colocando o resultado em EAX
Mov ECX, EAX	=> Copia o conteúdo do acumulador para outro registrador de uso geral (ex. ECX)
Mov EAX, D	=> Copia o conteúdo do endereço D para o acumulador (EAX)
Sub EAX, ECX	=> Subtrai o conteúdo acumulador (EAX) pelo conteúdo do registrador ECX e armazena o resultado em EAX
Mul D	=> Multiplica o conteúdo do acumulador pelo conteúdo do endereço D armazenando em EAX.
Add EAX, X	=> Adiciona ao acumulador (EAX) o conteúdo do endereço de memória X
Add EAX, A	=> Adiciona ao acumulador (EAX) o conteúdo do endereço de memória A
Mov X, EAX	=> Copia o conteúdo do acumulador para um endereço de memória X (Endereço de resposta).

OBS: Pode considerar como registradores: EAX de 32 bits, ou AX de 16 bits, ou AL (o RAX é apenas para processadores 64bits). Da mesma forma: EBX, BX, BL; ECX,CX, CL, etc.

Instruções escolhidas para PowerPC:

FONTE DE CONSULTA: PÁGINAS 52, 56 e 58 do manual:
http://download.boulder.ibm.com/ibmdl/pub/software/dw/library/es-ppcbook1.zip

add RT, RA, RB
adicionar o valor do registrador A com o valor do registrador B e armazena o resultado no registrador T
RT ← (RA) + (RB)

subf RT, RA, RB
subtrai o valor do registrador B com o valor do registrador A e armazena o resultado no registrador T
RT ← ¬(RA) + (RB) + 1 (soma feita por complem. a 2 invertendo o 1º. Operando)

mulld RT, RA, RB
multiplica o valor do registrador A com o valor do registrador B e armazena o resultado no registrador T

divid RT, RA, RB
divide o valor do registrador A com o valor do registrador B e armazena o resultado no registrador T

ld REGA, 0(REGB)
Usa o conteúdo do Registrador B (REGB) como endereço de memória do valor a ser carregado no registrador A (REGA). O 0 indica quantidade de bits, para nosso exercício, manteremos este valor.

std REGA, 0(REGB)
Usa o conteúdo do Registrador B (REGB) como endereço de memória do valor a ser salvo do registrador A (REGA). O 0 indica quantidade de bits, para nosso exercício, manteremos este valor.

li REG, VALUE
Carrega o registrador REG com o número VALUE

X= A+(B*(C-A)+(D-E/B)*D)

RESPOSTA: No PowerPc, os registradores são numerados de 0 a 31.

li 1, C - Carrega no reg. 1 o endereço C (não o conteúdo)
ld 2, 0(1) - Carrega no reg. 2 o conteúdo do endereço contido no reg. 1
li 1, A - Carrega no reg. 1 o endereço A (não o conteúdo)
ld 3, 0(1) - Carrega no reg. 3 o conteúdo do endereço contido no reg. 1
subf 2, 3, 2 - Carrega no reg. 2 o resultado da subtração do reg.2 pelo reg.3
li 1, B - Carrega no reg. 1 o endereço B (não o conteúdo)
ld 4, 0(1) - Carrega no reg. 4 o conteúdo do endereço contido no reg. 1
mulld 2, 4, 2 - Carrega no reg. 2 o conteúdo da multiplicação do reg.4 pelo reg. 2
li 1, E - Carrega no reg. 1 o endereço E (não o conteúdo)
ld 5, 0(1) - Carrega no reg. 5 o conteúdo do endereço contido no reg. 1
divid 5, 5, 4 - Carrega no reg. 5 o conteúdo da divisão do reg.5 pelo reg. 4
li 1, D - Carrega no reg. 1 o endereço D (não o conteúdo)
ld 6, 0(1) - Carrega no reg. 6 o conteúdo do endereço contido no reg. 1
subf 5, 5, 6 - Carrega no reg. 5 o resultado da subtração do reg.6 pelo reg.5
mulld 5, 5, 6 - Carrega no reg. 5 o conteúdo da multiplicação do reg.5 pelo reg. 6
add 5, 2, 5 - Carrega no reg. 5 o resultado da adição do reg.2 pelo reg.5
add 5, 3, 5 - Carrega no reg. 5 o resultado da subtração do reg.3 pelo reg.5
li 1, X - Carrega no reg. 1 o endereço X (não o conteúdo)

std 5, 0(1) - Armazena no endereço contido no reg. 1 o conteúdo do reg. 5

4. (0,5) Explique, comparando:

a) Computadores vetoriais e Computadores matriciais

O termo computadores vetoriais que correspondem a sistemas compostos por processadores vetoriais que freqüentemente são associados à organizações de ULAs com pipeline de operações.

E o termo computadores matriciais correspondem a sistemas compostos por processadores matriciais cuja organização é formada de ULAs paralelas.

b) Sistemas SMP e Sistemas NUMA

Sistemas SMP (ou UMA) têm como característica o acesso a todas as partes da memória principal com tempo de acesso uniforme. Em sistemas NUMA, todos os processadores possuem também acesso a todas as partes da memória principal podendo diferir o tempo de acesso em relação às posições da memória e processador.

Nos sistemas SMP o aumento no número de processadores tem como conseqüência problemas de tráfego no barramento comum degradando o desempenho. Uma solução para isto é a utilização de clusters, que tem, usualmente, como conseqüência alterações significativas na aplicação (software). Nos sistemas NUMA podemos ter vários nós multiprocessadores, cada qual com seu próprio barramento, resultando em pequenas alterações na aplicação (software).

c) Arquiteturas RISC e Arquiteturas CISC

RISC: Reduced Instruction Set Computer – Computador com um conjunto reduzido de instruções
CISC - Complex Instruction Set Computer: Computador com um conjunto complexo de instruções

CISC: Principais características:

Possui microprogramação para aumento da quantidade de instruções incluindo novos modos de endereçamento, de forma a diminuir a complexidade dos compiladores e em conseqüência permitir linguagens de alto nível com comandos poderosos para facilitar a vida dos programadores. Em contrapartida, muitas instruções significam muitos bits em cada código de operação, instrução com maior comprimento e maior tempo de interpretação.

RISC: Principais características:

Menor quantidade de instruções e tamanho fixo. Não há microprogramação. Permite uma execução otimizada, mesmo considerando que uma menor quantidade de instruções vá conduzir a programas mais longos. Uma maior quantidade de registradores e suas utilizações para passagem de parâmetros e recuperação dos dados, permitindo uma execução mais otimizada de chamada de funções. Menor quantidade de modos de endereçamento com o objetivo da reduzir de ciclos de relógio para execução das instruções. Instruções de formatos simples e únicos tiram maior proveito de execução com pipeline cujos estágios consomem o mesmo tempo.

5. (1,0) Considere uma máquina que utiliza *n* bits para representar dados. Em uma operação de soma de dois valores, considere os bits "vai-um" nas posições *n-1* (cn-1) e *n* (cn). Indique uma operação a ser realizada sobre estes bits "vai-um" de tal maneira que seja detectado estouro (*overflow)* considerando-se que os *n* bits representam:

a) inteiros sem sinal

O bit "vai-um" na posição *n* (cn) é igual a 1 quando ocorre estouro. Então basta comparar cn com 1, se for igual houve estouro.

b) inteiros em complemento a 2

O bit "vai-um" cn é diferente do bit "vai-um" cn-1 quando ocorre estouro. Então basta comparar estes dois bits e caso eles sejam diferentes houve estouro.

6. (1,0) Considere um computador, cuja representação para ponto fixo e para ponto flutuante utilize 16 bits. Na representação para ponto flutuante, o número 0 é representado pelo conjunto de bits $(0000)_{16}$. As outras combinações de bits representam números normalizados do tipo $+/-(1,b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{-7}b_{-8}b_{-9}b_{-10} \times 2^{Expoente})$, onde o bit mais à esquerda representa o sinal (0 para números positivos e 1 para números negativos), os próximos 5 bits representam o expoente em excesso de 16 e os 10 bits seguintes representam os bits b_{-1} a b_{-10}, como mostrado na figura a seguir:

S	Expoente representado em excesso de 16	$b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}\ b_{-5}\ b_{-6}\ b_{-7}\ b_{-8}\ b_{-9}\ b_{-10}$

a) (0,4) Considere o seguinte conjunto de bits representado em hexadecimal D000. Indique o valor deste número **em decimal**, considerando-se que o conjunto representa:

O número (D000)16 = (11010000 00000000)2

(a.1) um inteiro sem sinal

Temos: $2^{15} + 2^{14} + 2^{12} = 53248$

(a.2) um inteiro em sinal magnitude

Temos: $-(2^{14} + 2^{12}) = -20480$

(a.3) um inteiro em complemento a 2

Temos: $-2^{15} + (2^{14} + 2^{12}) = -12288$

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

11010000 00000000
Sendo:
Sinal = 1 => negativo
Expoente = 10100 = 20 – 16 => expoente = 4
Mantissa = 0000000000

Temos então => $-(1,00000)_2 \times 2^4 = -16$

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

b.1) +35,625

Convertendo para binário = 100011 ,101 => normalizando $1,00011101 \times 2^5$
Temos então:
Sinal = 0 (positivo)
Expoente = 5+16 = 21 = 10101
Mantissa = ,00011101
Resultado: 0 10101 0001110100 ou 01010100 01110100

b.2) –3,1

Convertendo para binário = 11 ,00011001100110011.... => normalizando $1,10001100110011001 \times 2^1$
Temos então:
Sinal = 1 (negativo)
Expoente = 1+16 = 17 = 10001
Mantissa = ,10001100110011001
Resultado: 1 10001 1000110011 ou 11000110 00110011

c) (0,2) Mostre a representação dos números dos itens da questão acima, caso se utilizasse a representação em complemento a 2 para representar o expoente.

c.1) +35,625 => normalizado $1,00011101 \times 2^5$
Temos então:
Sinal = 0 (positivo)

Expoente por C2 = 5 => C2 = 00101
Mantissa = ,00011101
Resultado: 0 00101 0001110100 ou 00010100 01110100

c.2) –3,1 => normalizado 1,10001100110011001 x 2^1
Temos então:
Sinal = 1 (negativo)
Expoente por C2 = 1 => C2 = 00001
Mantissa = ,10001100110011001
Resultado: 1 00001 1000110011 ou 10000110 00110011

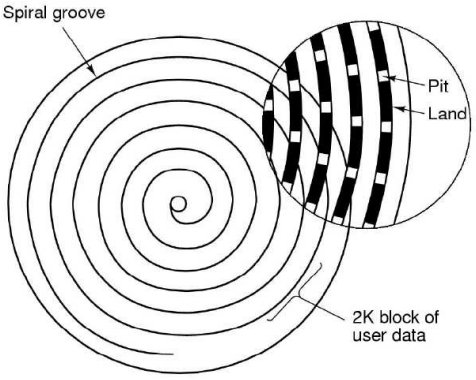
d) (0,2) Indique o menor e o maior valor positivo normalizado na representação em ponto flutuante para este computador. Mostre os valores **em decimal**.

Menor = 1 11111 1111111111 => $-(1,1111111111)_2$ x 2^{15} => -65504
Maior = 0 11111 1111111111 => $+(1,1111111111)_2$ x 2^{15} => +65504

7. (1,0) Explique como funciona um CD-ROM e um DVD (sugestões de fonte de consulta: livro do Stallings e sites na Internet. Na sua resposta indique as suas fontes de consulta).

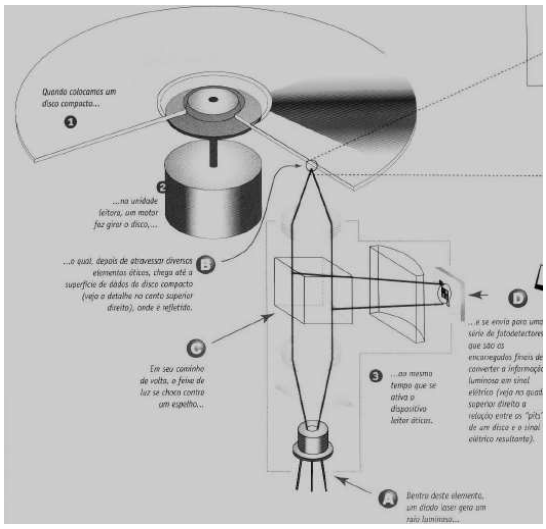
Fontes:
Arquitetura e Organização de Computadores – William Stallings
Organização estruturada de Computadores – Andrew Tanenbaum

CD-ROM (compact disc read-only-memory) – disco compacto de memória apenas de leitura. A principal diferença de um CD de áudio para CD-ROM é que este é mais resistente e possui mecanismo de correção de erros para assegurar que os dados sejam transferidos corretamente. O disco é constituído de uma resina do tipo policarbonato e revestido com uma superfície com alto índice de reflexão, normalmente de alumínio. A informação digital registrada é impressa nessa superfície refletiva (LANDs) como uma série de sulcos microscópicos (PITs). A gravação é feita primeiramente com um laser de alta intensidade muito bem focado para criar um disco matriz. Essa matriz é utilizada para fazer um molde para estampar cópias. A superfície sulcada das cópias é protegida contra pó e arranhões por uma cobertura de laca ou verniz clara.



A informação gravada em um CD-ROM é lida por um feixe de laser de baixa potência, acondicionando no dispositivo de leitura do tipo óptico.

Na reprodução, os CDs são iluminados por luz infravermelha com 0,78 mícron de comprimento de onda que incide sobre os pits e lands. O laser está do lado do policarbonato, fazendo



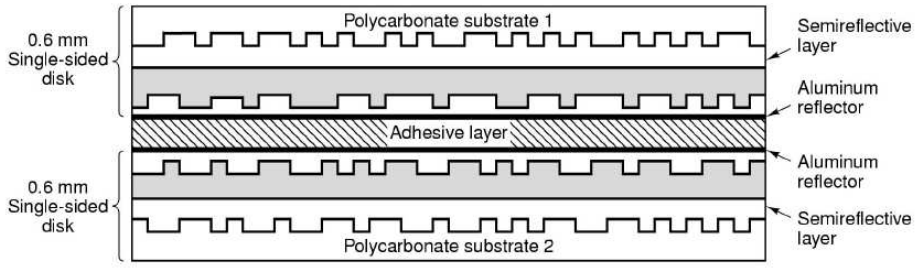
com que os pits se comportem nesse lado da mesma maneira que as saliências se comportam do lado contrário.

A luz refletida no pit em conjunto com a luz refletida na superfície é destrutiva e retornam menos luz para o dispositivo fotodetector do CD player do que a luz que sai de um land. Entretanto em vez de usar o pit para 0 e o land para 1, usa-se a transição de pit-land ou land-pit para representar o bit 1 e a ausência dessa transição de 0.

Os pits e os lands são escritos em uma única espiral contínua, começando próximo ao buraco do CD, espiral essa que se desenvolve em direção à borda do CD, com 32 mm de distância entre duas linhas. A espiral gera 22188 revoluções em torno do disco (600 por mm) isso equivale a 5,6 km.

DVD: originalmente uma abreviatura de Digital Vídeo Disk, mas agora oficialmente, Digital Versatile Disk. Os DVDs usam o mesmo projeto dos CD-ROMs, com discos de 120mm de policarbonato contendo pits e lands e que são iluminados por um diodo a laser e lidos por um fotodetector. A novidade está no uso de pits menores (0,4 mícron), uma pista espiral mais apertada (0,74 mícron), um laser vermelho e mais fino (0,65 mícron). Essas inovações fizeram com que a capacidade aumentasse sete vezes, alcançando a marca de 4,7GB. Um drive de DVD com velocidade de uma vez opera a 1,4MB/s

Formatos de DVD:
–Um único lado, uma única camada: 4,7GB
–Um único lado, duas camadas (8,5GB)
–Dois lados, uma única camada (9,4GB)
–Dois lados, duas camadas (17GB)



8. (1,0) Explique como funciona a tecnologia *plug and play* que significa “ligar e usar” (sugestões de fonte de consulta: existem vários sites na Internet que explicam esta tecnologia. Na sua resposta indique as suas fontes de consulta)

Fonte:

www.clubedohardware.com.br

Objetivo:

A tecnologia ‘’’Plug and Play (PnP)’’’ , que significa “ligar e usar”, foi criada com o objetivo de permitir que a BIOS ou sistema operacional instalem e configurem quaisquer combinações de placas de expansão e dispositivos. Desta forma, aquele método de configuração de jumpers ou chaves passa a ser substituído por um gerenciamento por software. A flexibilidade conseguida com este sistema é tal que permite reconfigurações mesmo em tempo de execução

A solução para o problema plug and play consiste em fazer com que o hardware, firmware, sistema operacional e aplicativos sejam capazes de compartilhar dispositivos através da definição de formatos de identificadores de dispositivos (tipo, de placa, fabricante, versão, etc.) e recursos exigidos (interrupção, canal de DMA e endereços de I/O ou memória), e que são estruturas de dados, padronizadas pela arquitetura plug and play.

Funcionamento:

O primeiro passo para que seja constituído o ambiente plug and play é identificar a configuração da máquina (feita inicialmente pela BIOS e posteriormente gerenciada pelo sistema operacional), ou seja, construindo-se uma árvore de

hardware. Tal árvore deve conter os registros necessários para estabelecerem-se a assinatura ou identificação dos dispositivos e a informação necessária para automaticamente localizar e carregar os device-drivers. Após carregar o device-driver, adiciona-se à árvore de hardware, mesmo em tempo de execução, um nó de dispositivo associado ao driver.

Vale observar que toda placa-mãe que suporta a arquitetura plug and play possui uma BIOS que busca, durante a inicialização do computador, e armazena os registros de nós de dispositivos, contendo as informações dos recursos exigidos. Isto orienta todas as partições de sistemas operacionais e os device-drivers que possam futuramente ser instalados. Após a instalação, os nós de dispositivos são acrescentados ou modificados segundo as necessidades do sistema operacional.

É importante ressaltar que quaisquer tipos de conexões elétricas fazem parte da árvore de hardware, inclusive todas as ligações hierárquicas entre os barramentos (ex: ponte barramento local - barramento ISA ou PCI).

Um problema surge ao distribuir os recursos pelos nós da árvore de hardware, de solução não trivial, que é o de identificar os dispositivos estáticos, como placas ISA que não têm a flexibilidade de alterar dinamicamente os recursos exigidos de interrupção, DMA e endereços de I/O ou memória. Por tal falta de flexibilidade, duas são as opções para tais dispositivos: mapeá-los de forma prioritária (ex: se o dispositivo exigir a IRQ 5, ele a terá disponível) ou desligá-los (desconectá-los logicamente da configuração da máquina e fisicamente do barramento).

A BIOS oferece também um conjunto padronizado de serviços referentes à arquitetura plug and play, como a obtenção dos recursos alocados para cada dispositivo. Tais serviços são acessados, por software, de duas formas: somente em modo real (ex: programas para DOS), através da interrupção por software 1Ah; e tanto em modo real quanto em modo protegido (ex: windows 95 e DOS), através de subrotinas (instrução call).

Dispositivos periféricos conectados diretamente à placa mãe do sistema (temporizadores, RAM CMOS, controladores de interrupção e DMA, interface de teclado e outros) e as placas ISA estáticas (que não são reconfiguráveis) têm seus recursos estrategicamente alocados como num PC AT convencional. Então, apenas os recursos restantes poderão vir a ser compartilhados pelas placas plug and play.

Comercialmente, são mais comuns placas plug and play para 3 tipos de barramentos comerciais: PCI, ISA e PCMCIA. Os dois primeiros casos serão comentados. Placas que satisfazem à especificação PCI já são, por natureza, plug and play. Tal especificação já reserva, para cada slot PCI lógico, 256 endereços de I/O consecutivos, onde, pelos primeiros 64, são feitas as programações de recursos e oferecidas informações outras, como identificações do fabricante e da versão e a localização da ROM de expansão.

Ao contrário do barramento ISA, podem existir múltiplos barramentos PCI numa configuração de máquina (a vantagem disto é descongestionar o fluxo de dados onde está conectada a CPU principal). Tais barramentos são hierarquizados da seguinte forma: existe um barramento raiz e os outros barramentos são interligados eletricamente por pontes PCI-PCI.

Uma placa ISA plug and play é uma variação nova dos adaptadores ISA, com a característica de admitir uma configuração dinâmica de recursos. Vale ressaltar que, para os dois casos, as pinagens dos conectores de encaixe de placas são idênticas. A diferença é que, na tradicional, a configuração é estabelecida por jumpers e chaves, e que, na plug and play, é por software, segundo padrões bem definidos.

Quanto a custos, as placas ISA tradicionais são de projeto e programação bem mais simples que as ISA plug and play, que, para atender aos padrões de configuração dinâmica, exigem circuitos digitais extras e uma programação bem mais complexa. Isto talvez explique o porquê de ainda projetarem-se bastante placas ISA tradicionais, principalmente no meio acadêmico ou mesmo no profissional em escala reduzida, isto quando a placa tem utilização bastante restrita.

9. (1,0) Explique como funcionam as três técnicas para realizar operações de entrada e saída: E/S programada, E/S dirigida por interrupção e acesso direto à memória. Indique pelo menos uma vantagem e uma desvantagem para cada uma delas.

a) 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 e utilização do processador para realizar a transferência de dados, o que também acarreta perda de ciclos de processador.

b) 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

c) 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 relacionadas a entrada e saída. A desvantagem é que precisamos de hardware adiciona