AD2 - Organização de Computadores 2010.2
Data de entrega 30/10/2010

"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. (1,0) Crie 3 conjuntos de instruções de um, dois, e três operandos, definidas em Linguagem Assembly, necessárias para a realização de operações aritméticas e elabore programas para o cálculo das equações abaixo (no total de 3 programas para cada item abaixo, sendo o 1º programa utilizando o conjunto de 1 operando, o 2º utilizando o conjunto de 2 operandos e finalmente o 3º utilizando o conjunto de 3 operandos). No conjunto de instruções proposto para dois e três operandos, projete instruções com endereçamento imediato, direto e indireto.

I_ CONJUNTO DE INSTRUÇÕES PARA 1 OPERANDO:

```
ADD X       => (ACC)<-(ACC)+(X)
SUB X       => (ACC)<-(ACC)-(X)
MUL X       => (ACC)<-(ACC)*(X)
DIV X       => (ACC)<-(ACC)/(X)
LOAD X      => (ACC)<- (X)
STORE X     => (X) <-(ACC)
```

II_ CONJUNTO DE INSTRUÇÕES PARA 2 OPERANDOS:

OBS: Em geral, para diferenciar as instruções conforme o modo de endereçamento, é acrescido ou modificado o mnemônico da instrução base. Nos casos abaixo, foram acrescidos as letras: i, d, n ; para indicar os modos Imediato, direto e indireto, respectivamente E caso estas mesmas instruções sejam utilizadas em um processador que possua também instruções de 1 e 3 operandos, uma nova identificação poderá ser acrescida ao rótulo da instrução para diferenciar das demais com 1 ou 3 operandos.

Imediato	Direto	Indireto

`ADDiD X,Y => (X)<-(X)+Y`
`SUBiD X,Y => (X)<-(X)-Y`
`MULiD X,Y => (X)<-(X)*Y`
`DIViD X,Y => (X)<-(X)/Y`
`MOViD X,Y => (X)<-Y`
 	
Neste modo, Y é um valor imediato
	`ADDdD X,Y => (X)<-(X)+(Y)`
	`SUBdD X,Y => (X)<-(X)-(Y)`
	`MULdD X,Y => (X)<-(X)*(Y)`
	`DIVdD X,Y => (X)<-(X)/(Y)`
	`MOVdD X,Y => (X)<-(Y)`
	
	Neste modo, Y é um registrador ou endereço de memória
		`ADDnD X,Y => (X)<-(X)+((Y))`
		`SUBnD X,Y => (X)<-(X)-((Y))`
		`MULnD X,Y => (X)<-(X)*((Y))`
		`DIVnD X,Y => (X)<-(X)/((Y))`
		`MOVnD X,Y => (X)<-((Y))`
		
		Neste modo, Y é um registrador ou endereço de memória que contenha a posição do valor desejado

III_ CONJUNTO DE INSTRUÇÕES PARA 3 OPERANDOS:

OBS: Em geral para diferenciar as instruções é acrescido ou modificado o mnemônico da instrução base. Nos casos abaixo, foram acrescidos as letras: i, d, n; para indicar os modos Imediato, direto e indireto, respectivamente. E caso estas mesmas instruções sejam utilizadas em um processador que possua também instruções de 1 e 2 operandos, uma nova identificação poderá ser acrescida ao rótulo da instrução para diferenciar das demais com 1 ou 2 operandos.

Imediato	Direto	Indireto

ADDiT X,Y,Z=>(X)<-(Y)+ Z
SUBiT X,Y,Z=>(X)<-(Y)- Z
MULiT X,Y,Z=>(X)<-(Y)+ Z
DIViT X,Y,Z=>(X)<-(Y)/ Z
 	
Neste modo, Z é um valor imediato
	ADDdT X,Y,Z=>(X)<-(Y)+(Z)
	SUBdT X,Y,Z=>(X)<-(Y)-(Z)
	MULdT X,Y,Z=>(X)<-(Y)*(Z)
	DIVdT X,Y,Z=>(X)<-(Y)/(Z)
	
	Neste modo, Z é um registrador ou endereço de memória
		ADDnT X,Y,Z=>(X)<-(Y)+((Z))
		SUBnT X,Y,Z=>(X)<-(Y)-((Z))
		MULnT X,Y,Z=>(X)<-(Y)*((Z))
		DIVnT X,Y,Z=>(X)<-(Y)/((Z))
		
		Neste modo, Z é um registrador ou endereço de memória que contenha a posição do valor desejado

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

PARA 1 OPERANDO:

```
LOAD A      =>    (ACC) <- (A)
DIV D       =>    (ACC) <- (ACC) / (D)
SUB C       =>    (ACC) <- (ACC) – (C)
STORE T1    =>    (T1)  <- (ACC)
LOAD E      =>    (ACC) <- (E)
DIV B       =>    (ACC) <- (ACC) / (B)
STORE T2    =>    (T2)  <- (ACC)
LOAD D      =>    (ACC) <- (D)
SUB T2      =>    (ACC) <- (ACC) – (T2)
LOAD C      =>    (ACC) <- (C)
ADD A       =>    (ACC) <- (ACC) + (A)
STORE T3    =>    (T3)  <- (ACC)
LOAD B      =>    (ACC) <- (B)
MUL T2      =>    (ACC) <- (ACC) * (T2)
MUL T3      =>    (ACC) <- (ACC) * (T3)
ADD E       =>    (ACC) <- (ACC) + (E)
ADD T1      =>    (ACC) <- (ACC) + (T1)
STORE X     =>    (X)  <- (ACC)
```

PARA 2 OPERANDOS:

```
MOVdD T1,A    =>    (T1) <- (A)
DIVdD T1,D    =>    (T1) <- (T1) / (D)
SUBdD T1,C    =>    (T1) <- (T1) – (C)
MOVdD T2,E    =>    (T2) <- (E)
DIVdD T2,B    =>    (T2) <- (T2) / (B)
MOVdD T3,D    =>    (T3) <- (D)
SUBdD T3,T2   =>    (T3) <- (T3) – (T2)
MOVdD T4,C    =>    (T4) <- (C)
ADDdD T4,A    =>    (T4) <- (T4) + (A)
MOVdD T2,B    =>    (T2) <- (B)
MULdD T2,T3   =>    (T2) <- (T2) * (T3)
MULdD T2,T4   =>    (T2) <- (T2) * (T4)
ADDdD T2,E    =>    (T2) <- (T2) + (E)
ADDdD T1,T2   =>    (T1) <- (T1) + (T2)
MOVdD X,T1    =>    (X)  <- (T1)
```

PARA 3 OPERANDOS:

```
DIVdT T1, A, D   =>  (T1) <- (A)  / (D)
SUBdT T1,T1, C   =>  (T1) <- (T1) - (C)
DIVdT T2, E, B   =>  (T2) <- (E)  / (B)
SUBdT T2, D,T2   =>  (T2) <- (D)  - (T2)
ADDdT T3, C, A   =>  (T3) <- (C)  + (A)
MULdT T2, B,T2   =>  (T2) <- (B)  * (T2)
MULdT T2,T2,T3   =>  (T2) <- (T2) * (T3)
ADDdT T2,T2, E   =>  (T2) <- (T2)  + (E)
ADDdT  X,T1,T2   =>  (X) <- (T1)  + (T2)
```

1.2. *Y= (A - B*(C + D/(E*(B-F))*C) + E)*

PARA 1 OPERANDO:

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

PARA 2 OPERANDOS:

```
MOVdD T1,B   =>   (T1) <- (B)
SUBdD T1,F   =>   (T1) <- (T1) – (F)
MULdD T1,E   =>   (T1) <- (T1) * (E)
MOVdD T2,D   =>   (T2) <- (D)
DIVdD T2,T1  =>   (T2) <- (T2) / (T1)
MULdD T2,C   =>   (T2) <- (T2) * (C)
ADDdD T2,C   =>   (T2) <- (T2) + (C)
MULdD T2,B   =>   (T2) <- (T2) * (B)
MOVdD  Y,A   =>   (Y)  <- (A)
SUBdD  Y,T2  =>   (Y)  <- (Y) – (T2)
ADDdD  Y,E   =>   (Y)  <- (Y) + (E)
```

PARA 3 OPERANDO:

```
SUBdT T1, B, F    =>  (T1) <- (B)   –  (F)
MULdT T1, E,T1    =>  (T1) <- (E)   *  (T1)
DIVdT T1, D,T1    =>  (T1) <- (D)   /  (T1)
MULdT T1,T1, C    =>  (T1) <- (T1)  *  (C)
ADDdT T1, C,T1    =>  (T1) <- (C)   +  (T1)
MULdT T1, B,T1    =>  (T1) <- (B)   *  (T1)
```

```
SUBdT T1, A,T1    =>  (T1) <- (A)   -  (T1)
ADDdT  Y,T1, E    =>  (Y)  <- (T1)  +  (E)
```

2. (1,0) Faça uma busca na lista dos 500 sistemas de computadores com melhor desempenho do mundo em http://www.top500.org e descreva o sétimo colocado (pesquise neste mesmo site e na internet).

Informações obtidas no site do Top 500 – lista publicada em jun/2010

7o. Colocado no Top500

Nome: Tianhe-1
Local: National SuperComputer Center em Tianjin / China
Modelo: NUDT TH-1 Cluster
Características: Cluster NUDT TH-1
Processadores: Total de 71.680 cores
CPU: Intel Xeon E5540/E5450 2530 MHz (10.12 GFlops)
GPU: ATI Radeon HD 4870-2
Interconexão: Infiniband
Fabricante: NUDT
Memória: cerca de 96 TB
Ano de instalação: 2009
Sistema Operacional: Linux
Capacidade de processamento máximo: cerca de 563 Tflops (trilhões de operações de ponto flutuante por segundo)

Texto retirado do site inovaçãotecnológica.com.br publicado em 29/10/2010 (http://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo=supercomputador-mais-rapido-mundo-china&id=010175101029)

“ A China apresentou aquele que deve ser o supercomputador mais rápido do mundo.

Segundo o Centro Nacional de Supercomputação, localizado na cidade de Tianjin, o Tianhe-1 tem uma velocidade sustentada de 2.507 trilhões de operações de ponto flutuante por segundo, ou 2,507 petaflops.

Teoricamente, o supercomputador, cujo nome significa Via Láctea, é capaz de atingir um pico de 4,7 petaflops.”

informações adicionais também retiradas do site acima citado:

“...*O Tianhe-1 é um supercomputador híbrido, construído com uma mistura de processadores Intel (CPUs) e processadores gráficos (GPUs) fabricados pela Nvidia.*

Segundo o blog da Nvidia, se o Tianhe-1 fosse construído usando apenas CPUs normais seriam necessários mais de 50.000 processadores, consumindo nada menos do que 12 megawatts de potência.

O uso de um modelo heterogêneo, mesclando GPUs paralelas com CPUs multicore, é possível obter mais desempenho e mais eficiência energética.

Com seus 6.144 processadores Intel e 5.120 GPU Nvídia, o novo supercomputador chinês consome "apenas" 4 megawatts, ou seja, três vezes mais eficiente em termos energéticos do que uma máquina que usasse apenas CPUs....

…

O sistema de rede criado pelos chineses consegue transferir dados com o dobro da velocidade da tecnologia de interconexão chamada InfiniBand, usada na maioria dos supercomputadores ao redor do mundo

…

Segundo a Universidade Nacional de Tecnologias de Defesa, da China, o Via Láctea tem 103 gabinetes do tamanho de uma geladeira cada um, pesa 155 toneladas e ocupa uma área de 1.000 metros quadrados.

Os cientistas chineses não consideram esta uma máquina pronta. Eles pretendem agora começar a adicionar 'centenas ou milhares' de processadores chineses, chamados Feiteng-1000, que deverão aumentar ainda mais o desempenho da máquina.

A China tem um projeto de longo prazo para fabricar CPUs que rivalizem com os processadores da Intel. 'Eles ainda não conseguiram, mas deverão chegar lá em um ano ou dois,' disse Dongarra em uma entrevista ao jornal New York Times."

3. (1,0) Responda as questões abaixo:

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

MODO DE ENDEREÇAMENTO	*DEFINIÇÃO*	*VANTAGENS*	*DESVANTAGENS*	*DESEMPENHO*
Imediato	*O campo operando contém o dado*	*Rapidez na execução da instrução*	*Limitação do tamanho do dado. Inadequado para o uso com dados de valor variável*	*Não requer acesso a memória principal. Mais rápido que o modo direto*
Direto	*O campo operando contém o endereço do dado*	*Flexibilidade no acesso a variáveis de valor diferente em cada execução do programa*	*Perda de tempo, se o dado é uma constante*	*Requer apenas um acesso a memória principal. Mais rápido que o modo indireto*
Indireto	*O campo operando corresponde ao endereço que contém a posição onde está o conteúdo desejado,*	*Manuseio de vetores (quando o modo indexado não está disponível). Usar como "ponteiro"*	*Muitos acessos à MP para execução*	*Requer 2 acessos a memória principal*

3.2. Qual é o objetivo do emprego do modo de endereçamento base mais deslocamento? Qual é a diferença de implementação e utilização 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 relocação de programa), bastando para isso uma única alteração no registrador base.

O base mais deslocamento tem como característica o endereço ser obtido da soma do deslocamento com o registrador base, diferindo do modo indexado onde o do registrador base é fixo e variar no deslocamento, ao contrário deste onde o deslocamento é fixo e com a alteração do registrador base permite-se a mudança do endereço.

Exemplos de instruções modo indexado:
LDX Ri, Op ==> ACC <--- ((Op) + (Ri))
ADX Ri, Op ==> ACC <--- ACC + ((Op) + (Ri))

Exemplo: instrução base mais deslocamento:
LDB Rb, Op ==> (ACC) <--- ((Op) + (Rb))
ADB Rb, Op ==> ACC <--- ACC + ((Op) + (Rb))

Sendo,

Op = Operando
Ri = Registrador de índice
Rb = Registrador base

4. (1,0) Explique, comparando:

4.1. Compilação e Interpretação (Dê exemplos de linguagens que se utilizem de compiladores e de linguagens que se utilizem de interpretadores).

A compilação 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).

Na interpretação cada comando do código fonte é lido pelo interpretador, convertido em código executável e imediatamente executado antes do próximo comando.

A interpretação tem como vantagem sobre a compilação a capacidade de identificação e indicação de um erro no programa-fonte (incluindo erro da lógica do algoritmo) durante o processo de conversão do fonte para o executável.

A interpretação tem como desvantagem o consumo de memória devido ao fato de o interpretador permanecer na memória durante todo o processo de execução do programa. Na compilação o compilador somente é mantido na memória no processo de compilação e não utilizado durante a execução. Outra desvantagem da interpretação está na necessidade de tradução de partes que sejam executadas diversas vezes, como os loops que são traduzidos em cada passagem. No processo de compilação isto só ocorre uma única vez. Da mesma forma pode ocorrer para o programa inteiro, em caso de diversas execuções, ou seja, a cada execução uma nova interpretação.

Exemplos de linguagem interpretadas: ASP, BASIC, Java, PHP, Python, Lisp entre outras.

Exemplos de linguagem compilada: C, Pascal, Delphi, Visual Basic, entre outras.

4.2. Sistemas SMP e Sistemas NUMA (Forneça exemplos atuais de 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 podem-se ter vários nós multiprocessadores, cada qual com seu próprio barramento, resultando em pequenas alterações na aplicação (software).

Exemplos de NUMA: HP superDome, Unisys ES7000

Exemplos de SMP: IBM p690 Regatta

4.3. Arquiteturas RISC e Arquiteturas CISC (Forneça exemplos em processadores atuais de características RISC e 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

Exemplos: IBM /370-168 (exemplo antigo) , Intel 80486 , Intel Pentium, Intel Xeon

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 redução 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.

Exemplos: Power PC, Série Sparc, Alpha 21064

5. (1,0) Descreva o processador Intel core I7, detalhando seu conjunto de instruções e modos de endereçamento. Pesquise na wikipedia e nos sites da Intel.

a) Conjunto de instruções para o Core I7

Fonte: Intel® 64 and IA-32 Architectures Software Developer's Manual (Manual para desenvolvedores de software para arquiteturas Intel-64 e IA-32), encontrado em http://www.intel.com/Assets/PDF/manual/253665.pdf

O processador I7 utiliza o mesmo conjunto de instruções de seu antecessor (Core 2 duo) acrescido do conjunto complementar de instruções: SSE 4.2

O conjunto de instruções do I7 é composto de:

- *Instruções de uso geral*
- *Instruções x87 FPU (instruções que operam com operandos em ponto flutuante.*
- *Extensões SIMD*
    - *MMX*
    - *SSE*
    - *SSE2*
    - *SSE3 (incluíndo o subgrupo SSSE3)*
    - *SSE4 (incluíndo os subgrupos SSE4.1 e SSE4.2), sendo o I7 o primeiro modelo de processador a utilizar o subgrupo SSE4.2*

Instruções de uso geral:

As instruções de uso geral consistem basicamente de : movimentação de dados, aritmética, lógica, fluxo do programa e operações de cadeia de caracteres que normalmente os programadores usam para escrever software para rodar em processadores que tenham o padrão de instruções Intel 64 e IA-32. Eles operam com dados contidos na memória, com registradores de uso geral (EAX, EBX, ECX, EDX, EDI, ESI, EBP e ESP) e regisradores EFLAGS. Eles também operam com as informações contidas nos endereços de memória, nos registradores de propósito geral, e nos registradores de segmento (CS, DS, SS, ES, FS e GS).

Este conjunto de instruções inclui a transferência de dados, aritmética binária de inteiros, aritmética decimal, operações lógicas, deslocamento e rotação, operações de bits e bytes, controle do programa, cadeias de caracteres, controle de flags, operações com registradores de segmento e miscelânea.

Instruções x87 FPU

São executados por FPU (Float Processor Unit – Unidade de processamento de ponto flutuante, ou seja, o coprocessador) do processador x87. Estas instruções operam com operandos no formato de ponto flutuante, inteiro, decimal e binário codificado (BCD). Essas instruções são divididas nos seguintes subgrupos: transferência de dados, carregamento de constantes e instruções de controle de FPU.

Extensões SIMD

Quatro extensões foram introduzidas na arquitetura IA-32 permitindo que processadores executem instruções que trabalhem com multiplos dados (SIMD). Estas extensões incluem a tecnologia MMX e extensões SSE, SSE2 e SSE3.

***Instruções MMX**:*

Instruções MMX operam com operandos inteiros do formato byte, doubleword, ou quadword contidos na memória, nos registradores MMX e / ou nos registradores de uso geral.

As. instruções MMX são divididas nos seguintes subgrupos: transferência de dados, conversão aritmética, comparação, operações lógicas, deslocamento e rotação, e instruções de gerenciamento de estado.

Extensão SSE,

Introduzidas inicialmente no Pentium III, foram adicionados oito novos registradores de 128 bits, conhecido como XMM0 a XMM7. As extensões AMD64 da AMD (originalmente chamado de x86-64 e depois repetido pela Intel) adicionaram mais oito, de XMM8 até XMM15.

Instruções SSE são divididos em quatro subgrupos

- *Instruções SIMD de ponto flutuante que operam com os registradores XMM*
- *Instruções de gerenciamento do estado MXSCR*
- *Instruções SIMD de inteiro de 64bits que operam sobre os registradores XMM*
- *Controle de cacheabilidade, prefetch e instrução de ordenação de instruções*

Extensão SSE2

Introduziada com o Pentium IV, operam com operandos de ponto flutuante com dupla precisão e operandos inteiros de formato: byte, doubleword e quadword armazenados nos registradores XMM.

Estas instruções são divididas em quatro subgrupos:

- *Instruções de ponto-flutuante com dupla precisão*
- *Instruções de conversão para ponto flutuante de simples precisão*
- *Instruções SIMD com formato interiro de 128bits*
- *Controle de cacheabilidade, prefetch e instrução de ordenação de instruções*

Extensão SSE3

A extensão SSE3 oferece 13 instruções que aceleram o desempenho da capacidade matemática do SSE, SSE2 e x87 FPU. Estas instruções podem ser agrupadas nas seguintes categorias:

- *Uma instrução x87FPU usado na conversão de formato inteiro*
- *Uma instrução SIMD de inteiro que endereça carga de dados não alinhados*
- *Duas instruções SIMD de ADD / SUB de ponto flutuante*
- *Quatro instruções SIMD de ADD / SUB horizontal de ponto flutuante*
- *Três instruções SIMD de LOAD / MOVE / DUPLICATE de ponto flutuante*
- *Duas instruções para sincronização de threads*

O Subgrupos SSSE3 fornece 32 instruções (representadas por 14 mnemônicos) para acelerar os cálculos com números inteiros. Estes incluem:

- *Doze instruções que realizam operações de adição ou subtração horizontal.*
- *Seis instruções que avaliam valores absolutos.*
- *Duas instruções que realizam operações de multiplicação e adição de forma a acelerar o avaliação de produtos de ponto flutuante.*

- *Duas instruções que aceleram operações de multiplicação no formato inteiro e produzir valores inteiros com escamação.*
- *Duas instruções que executam um byte-wise, shuffle no local indicado pelo operando de controle de shuffle.*
- *Seis instruções que negativam inteiros no operando destino se o sinal do elemento correspondente ao operando de origem é menor que zero.*
- *Duas instruções que alinham dados de uma composição de dois operandos.*

Extensão SSE4

Intel ® Streaming SIMD Extensions 4 (SSE4) apresenta 54 novas instruções classificadas nos seguintes subgrupos:

- *SSE 4.1: 47 instruções*
- *SSE 4.2 7 instruções (inicialmente lançadas para os I7).*

O SSE4.1 é voltado para melhorar o desempenho dos meios de comunicação, imagem e cargas de trabalho 3D. SSE4.1 acrescenta instruções que melhoram significativamente a compilação e vetorização. Aumenta o auxilio no processamento do formato dword.

As 47 instruções do SSE4.1 incluem:

• Duas instruções que realizam multiplicação no formato dword.
• Duas instruções de ponto flutuante do tipo dot products com seleção de entrada / saída
• Uma instrução que realiza cargamento com streaming hint.
• Seis instruções simples de conversão de formatos.
• Oito instruções de expansão de suporte para formatos inteiros MIN / MAX.
• Quatro instruções com suporte ao arrendondamento de ponto flutuante com o modo de arredondamento selecionável..
• Sete instruções que otimizam a inserção e extração de dados de registos XMM
• Doze instruções para conversão de formatos de inteiro (sinal e extensão zero).
• Uma instrução de otimização SAD (soma de diferença absoluta) para geração de blocos de pequeno tamanhos
• Uma instrução auxiliar em operações de busca horizontal.
• Uma instrução otimizada para comparações mascaradas.
• Uma instrução adiciona comparação de igualdade para formato qword .
• Uma instrução adiciona satuação sem sinal para formato dword;

Os sete instruções SSE4.2 incluem:

- *Cinco das sete instruções SSE4.2 podem usar um registrador XMM como origem ou destino. Nestas estão inclusas 4 instruções para processamento de texto/string e uma instrução SIMD para comparações no formato quadword.*
- *As 2 instruções restantes utilizam registradores de uso geral para melhorar o desempenho no processamento de funções em áreas específicas.*

b) Modos de endereçamento do I7 seguem o padrão Intel x86 também presente no Pentium II.
Fonte: livro **Arquitetura e Organização de Computadores, de William Stallings**

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).

6. (1,6) Considere as seguintes variáveis com as atribuições de valores decimais:
   A= -14;
   B=-2;
   C=+15;
   D=+12

   6.1. Considere uma máquina que utiliza 5 bits para representar inteiros com sinal em sinal e magnitude e não detecta quando ocorre estouro nos valores atribuídos às variáveis.

   6.1.1. (0,2) Indique a representação dos valores atribuídos nesta máquina às variáveis A, B, C e D

   $A = -14 \Rightarrow 11110_2$
   $B = -2 \Rightarrow 10010_2$

$C = +15 \Rightarrow 01111_2$
$D = +12 \Rightarrow 01100_2$.

6.1.2. (0,6) Considere que as seguintes atribuições foram executadas nesta máquina:

E=A+B;
F=C-B;
G=C+D;
H=A+D;
Indique o valor em **decimal** atribuído às variáveis **E, F, G e H.**

$E = A + B \Rightarrow 11110 + 10010 = 10000 \Rightarrow 0_{10}$ (esperado -16)
$F = C - B \Rightarrow 01111 - 10010$ (torna negativo o segundo termo)
$= 01111 + 00010 = 00001 \Rightarrow +1_{10}$ (esperado +17)
$G = C + D \Rightarrow 01111 + 01100 = 01011 \Rightarrow +11_{10}$ (esperado +27)
$H = A + D \Rightarrow 11110 + 01100 = 10010 \Rightarrow -2_{10}$ (esperado -2)

6.2. Considere uma máquina que utiliza 5 bits para representar inteiros com sinal em complemento a 2 e não detecta quando ocorre estouro nos valores atribuídos às variáveis.

6.2.1. (0,2) Indique a representação dos valores atribuídos nesta máquina às variáveis A, B, C e D.

$A = -14 \Rightarrow 10010_2$
$B = -2 \Rightarrow 11110_2$
$C = +15 \Rightarrow 01111_2$
$D = +12 \Rightarrow 01100_2$

6.2.2. (0,6) Considere que as seguintes atribuições foram executadas nesta máquina:
E=A+B;
F=C-B;
G=C+D;
H=A+D;
Indique o valor em **decimal** atribuído às variáveis E, F, G e H.

$E = A + B \Rightarrow 10010 + 11110 = 10000 \Rightarrow -16_{10}$ (esperado -16)
$F = C - B \Rightarrow 01111 - 11110$ (torna negativo o segundo termo)
$= 01111 + 00010 = 10001 \Rightarrow -15_{10}$ (esperado +17)
$G = C + D \Rightarrow 01111 + 01100 = 11011 \Rightarrow -5_{10}$ (esperado +27)
$H = A + D \Rightarrow 10010 + 01100 = 11110 \Rightarrow -2_{10}$ (esperado -2)

7. (1,8) Considere um computador, cuja representação para ponto fixo e para ponto flutuante utilize 18 bits.

7.1. (0,6) Considere o seguinte conjunto de bits representado em hexadecimal 30E00. Indique o valor deste número **em decimal**, considerando-se que o conjunto representa:

$(30E00)_{16}$ $(110000111000000000)_2$

7.1.1. um inteiro sem sinal
$2^{17} + 2^{16} + 2^{11} + 2^{10} + 2^0 = 200.192$

7.1.2. um inteiro em sinal magnitude
$-(2^{16} + 2^{11} + 2^{10} + 2^0) = -69.120$

7.1.3. um inteiro em complemento a 2
$-2^{17} + 2^{16} + 2^{11} + 2^{10} + 2^0 = -61.952$

7.2. Na representação em ponto flutuante, como na representação IEEE 754, utiliza-se o bit mais à esquerda para representar o sinal, os próximos 6 bits representam o expoente e os 11 bits seguintes representam os bits depois da vírgula. Quando todos os bits que representam o expoente são iguais a 0 ou iguais a 1 temos os casos especiais. Caso contrário, as combinações possíveis de bits representam números normalizados no formato $+/-(1,b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{7}b_{8}b_{-9}b_{-10}b\text{-}_{11} \times 2^{\text{Expoente}})$, 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 excesso e os 11 bits seguintes representam os bits b_{-1} a b_{-11}, como mostrado na figura a seguir:

S	Expoente representado em excesso	$b_{-1}\, b_{-2}\, b_{-3}\, b_{-4}\, b_{-5}\, b_{-6}\, b_{-7}\, b_{-8}\, b_{-9}\, b_{-10}\, b_{-11}$
1	6	11

7.2.1. (0,3) Determine o valor do excesso utilizado, sabendo que os projetistas desta máquina utilizaram o mesmo critério utilizado pelo padrão IEEE754 para definir o valor do excesso.

Excesso $= 2^{n-1} - 1 = 2^{6-1} - 1 = 31$

7.2.2. (0,3) Indique o valor do conjunto de bits do item 7.1 considerando que este conjunto está representando um número normalizado em ponto flutuante com a representação acima.

1 100001 11000000000
Sendo:
Sinal = 1 => negativo
Expoente = 100001 = 33 – 31 => expoente = 2
Mantissa = 11000000000

Temos então => - 1, 11000000000 x $2^2 = (-111)_2 = (-7)_{10}$

7.2.3. (0,2) Qual será a representação em ponto flutuante dos seguintes valores decimais neste computador:

7.2.3.1. +10,75

+10,75 = 1010,11 = 1,01011 x 2^3
Sinal = 0 = positivo
Expoente = 3 +31 = 34 = 100010 (excesso de 31)
Mantissa = 01011
Resposta: 0 100010 01011000000

7.2.3.2. –2,1

-2,1 = 10,00011001100110011... = 1,00001100110011001100110011... x 2^{+1}
Sinal = 1 = negativo
Expoente = +1 + 31 = 32 = 100000 (excesso de 31)
Mantissa = 00001100110011001100110011...
Resposta: 1 100000 00001100110

7.2.4. (0,4) Indique o menor e o maior valor positivo **normalizado** na representação em ponto flutuante para este computador. Mostre os valores **em decimal**.

Menor valor positivo:
0 000001 00000000000 => 1,00000000000 x $2^{-30} = +0,93132257 \times 10^{-9}$

Maior valor positivo:
0 111110 11111111111 => 1,11111111111 x $2^{+31} = +4.293.918.720$

8. (0.8) Explique o funcionamento do mouse ótico. (sugestões de fonte de consulta: livro do Stallings e do Mário Monteiro e o site http://computer.howstuffworks.com. Na sua resposta indique as suas fontes de consulta).

Fonte: http://computer.howstuffworks.com *acessado em 16/10/2010*

Desenvolvido pela Agilent Technologies e lançado ao mundo no final de 1999, o mouse óptico utiliza uma câmara minúscula para captar milhares de imagens a cada segundo. Capaz de trabalhar em praticamente qualquer superfície sem um mouse pad. O mouse óptico utiliza uma pequena luz vermelha de diodos emissores de luz (LED), que é refletida na superfície e captada por um semicondutor de complemento de óxido metálico (CMOS). Além dos LEDs, mouses ópticos inovadores utilizam o laser a fim de detectar mais detalhes na superfície em comparação com a tecnologia LED.

Abaixo segue a descrição do funcionamento do mouse óptico:

- *O sensor CMOS envia cada imagem para um processador de sinal digital (DSP). O DSP detecta padrões em uma imagem e examina como os padrões se moveram desde a imagem anterior.*
- *Com base na mudança de padrões sobre uma seqüência de imagens, o DSP determina o quanto o mouse se moveu e envia as coordenadas correspondentes para o computador.*
- *O computador move o cursor na tela baseado nas coordenadas recebidas pelo mouse. Isto acontece centenas de vezes por segundo.*

Vantagens dos mouses ópticos sobre os antecessores, os de esfera:

- *Sem partes móveis significa menos desgaste e menor chance de falha.*
- *Não há como entrar poeira dentro do mouse, de forma a interferir nos sensores de monitoramento.*
- *Aumento da resolução de rastreamento obtem-se uma resposta mais suave.*
- *Não é necessário uma superfície especial, como um mouse pad.*

9. (0,8) 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.

E/S programada: *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. A vantagem deste método é: hardware simples. 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 Acesso Direto à Memória (ADM): *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 ADM 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 ADM 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 ADM, o processador pode continuar executando outras instruções. O controlador de ADM 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 adicional.*