Assembly Language for Intel-Based Computers, 5th Edition

Kip R. Irvine

Processamento Instruções em Ponto-Flutuante

Slide show prepared by the author

Revision date: June 4, 2006

índice

- **Representação binária de ponto-flutuante**
- Unidade de ponto-flutuante

Ariane 5

Ariane 5

- Explodiu 37 segundos após decolagem
- Custo: U$ 7 Bilhões
- Tempo de Desenvolvimento: 10 anos
- Por que?
  - Foi computada a velocidade horizontal como número em ponto flutuante de 64 bits
  - Convertido para inteiro de 16-bits
  - Funcionou bem para o Ariane 4
  - Ocorreu overflow para o Ariane 5 (mesmo software)

- **Exemplos de desastres devido a cálculos numéricos errados:**
  http://www.ima.umn.edu/~arnold/disasters/

Representação binária de ponto-flutuante

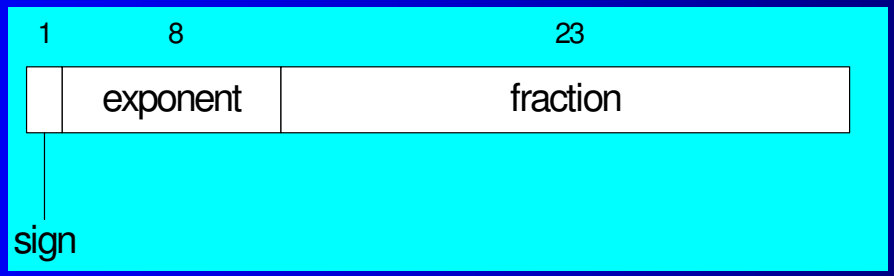
- Reais binários ponto-flutuante IEEE
- Expoente
- Números ponto-flutuante binários normalizados
- Criando a representação IEEE
- Convertendo frações decimais em reais binários

Reais binários ponto-flutuante IEEE-754

- Tipos
  - Precisão simples
    - 32 bits: 1 bit de sinal, 8 bits de expoente e 23 bits para a parte fracionária do significando.
  - Precisão dupla
    - 64 bits: 1 bit de sinal, 11 bits de expoente e 52 para a parte fracionária do significando.
  - Precisão dupla estendida
    - 80 bits: 1 bit de sinal, 16 bits de expoente e 63 bits para a parte fracionária do significando.

Formato de precisão simples

Intervalo normalizado aproximado: 2^{-126} a 2^{127}.
também chamado de *short real*.



Componentes de um real de precisão simples

- Sinal
  - 1 = negativo, 0 = positivo
- Significando
  - Dígitos decimais à esquerda e à direita do ponto decimal
  - Notação posicional ponderada
  - Exemplo:

    $123.154 = (1 \times 10^{2}) + (2 \times 10^{1}) + (3 \times 10^{0}) + (1 \times 10^{-1}) + (5 \times 10^{-2}) + (4 \times 10^{-3})$
- Expoente
  - Inteiros sem sinal
  - Polarização (bias) (127 para precisão simples)

Frações decimais vs ponto-flutuante binário

Binary Floating-Point	Base 10 Fraction
11.11	3 3/4
101.0011	5 3/16
1101.100101	13 37/64
0.00101	5/32
1.011	1 3/8
0.00000000000000000000001	1/8388608

Table 17-3 Binary and Decimal Fractions.

Binary	Decimal Fraction	Decimal Value
.1	1/2	.5
.01	1/4	.25
.001	1/8	.125
.0001	1/16	.0625
.00001	1/32	.03125

Expoente

- Exemplos de expoentes representados em binário
- somar 127 ao expoente para produzir o expoente polarizado (biased)

Exponent (E)	Biased (E + 127)	Binary
+5	132	10000100
0	127	01111111
-10	117	01110101
+127	254	11111110
-126	1	00000001
-1	126	01111110

Números ponto-flutuante binários normalizados

- A mantissa é normalizada quando um 1 aparece à esquerda do ponto binário
- Não-normalizado: o ponto binário pode variar até que o expoente seja zero
- Exemplos

Unnormalized	Normalized
1110.1	1.1101×2^{3}
.000101	1.01×2^{-4}
1010001.	1.010001×2^{6}

Codificação de números reais

- Números finitos normalizados
  - Todos os valores finitos não-zeros que podem ser codificados com número real normalizado entre zero e infinito
- Infinitos positivo e negativo
- NaN (not a number)
  - Padrão de bits que não é um valor válido FP
  - Dois tipos:
    - quiet
    - signaling

Codificação de números reais *(cont)*

- Codificações específicas (precisão simples):

Value	Sign, Exponent, Significand
Positive zero	0 00000000 00000000000000000000000
Negative zero	1 00000000 00000000000000000000000
Positive infinity	0 11111111 00000000000000000000000
Negative infinity	1 11111111 00000000000000000000000
QNaN	x 11111111 1xxxxxxxxxxxxxxxxxxxxxx
SNaN	x 11111111 0xxxxxxxxxxxxxxxxxxxxxx[a]

Exemplos (precisão simples)

- Ordem: bit de sinal, bits de expoente e parte fracionária (mantissa)

Binary Value	Biased Exponent	Sign, Exponent, Fraction
-1.11	127	1 01111111 11000000000000000000000
+1101.101	130	0 10000010 10110100000000000000000
-.00101	124	1 01111100 01000000000000000000000
+100111.0	132	0 10000100 00111000000000000000000
+.0000001101011	120	0 01111000 10101100000000000000000

Convertendo frações para reais binários

- Expressar como uma soma de frações tendo denominadores que são potências de 2
- Exemplos

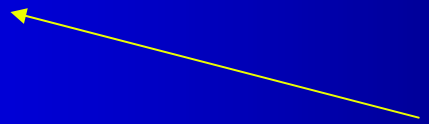
Decimal Fraction	Factored As...	Binary Real
1/2	1/2	.1
1/4	1/4	.01
3/4	1/2 + 1/4	.11
1/8	1/8	.001
7/8	1/2 + 1/4 + 1/8	.111
3/8	1/4 + 1/8	.011
1/16	1/16	.0001
3/16	1/8 + 1/16	.0011
5/16	1/4 + 1/16	.0101

Convertendo precisão simples para decimal

1. Se o MSB é 1, o número é negativo; caso contrário, positivo.
2. Os seguintes 8 bits representam o expoente. Subtrair o binário 01111111 (decimal 127), produzindo o expoente não-polarizado. Converter o expoente não-polarizado para decimal.
3. Os 23 bits seguintes representam o significando. Incluir "1.", seguido dos bits de significando. Zeros à direita podem ser ignorados.
4. Escalar o significando produzido no passo 3, deslocando o ponto binário o número de vezes igual ao valor do expoente. Deslocar à direita se o expoente é positivo, ou à esquerda se o expoente for negativo.
5. Converter a representação real binária produzida no passo 4 para representação decimal, e finalmente incluir o sinal.

Exemplo

Converter 0 10000010 01011000000000000000000
para Decimal

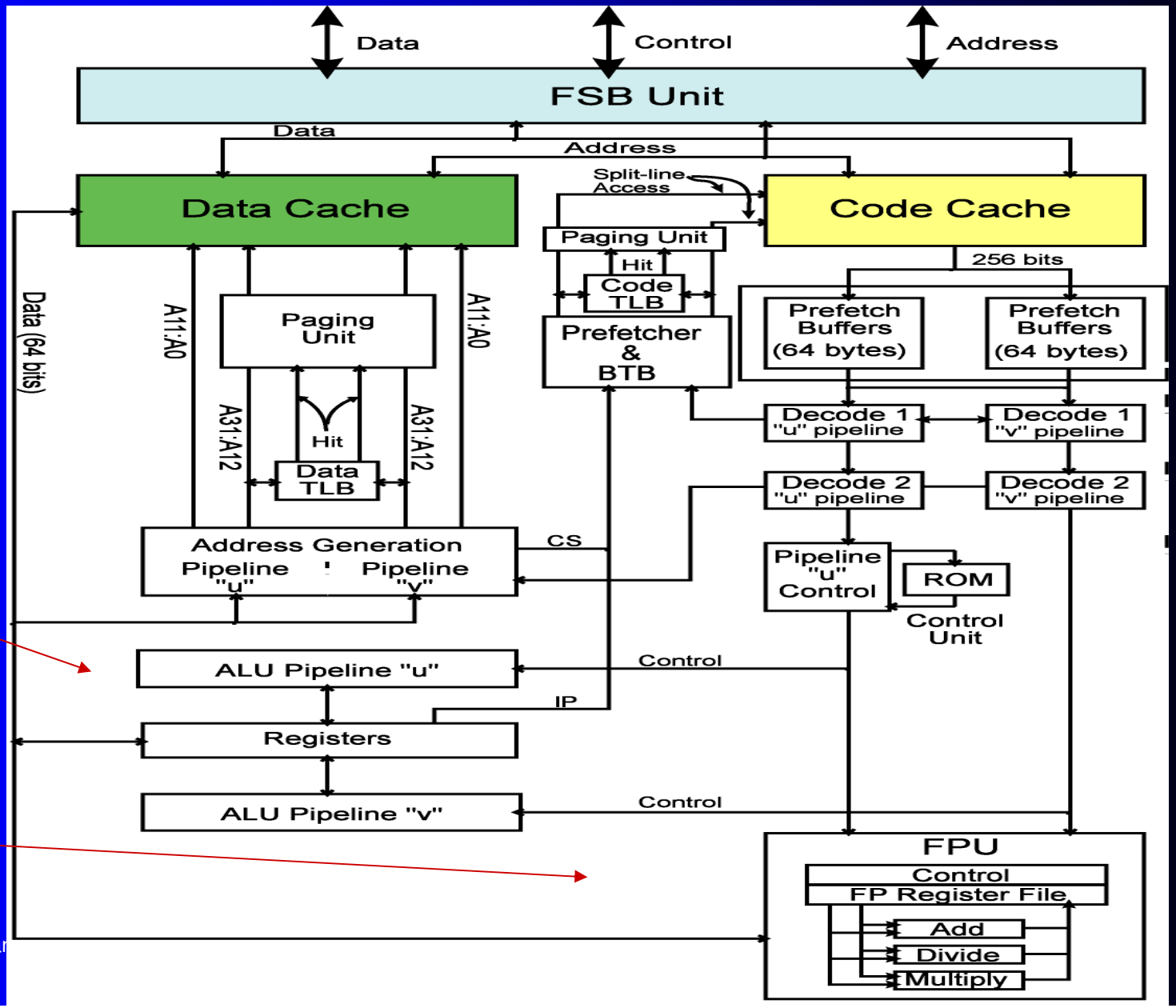


1. O número é positivo.
2. O expoente não-polarizado é o binário 00000011, ou decimal 3.
3. O significando é dado por 1. + 01011 = 1.01011
4. O número real binário é +1010.11
5. O valor decimal é +(10 + 3/4)= +(10.75)

Seção seguinte

- **Representação binária de ponto-flutuante**
- Unidade de ponto-flutuante

Unidade de ponto-flutuante (FPU)



Procesamento de Inteiros

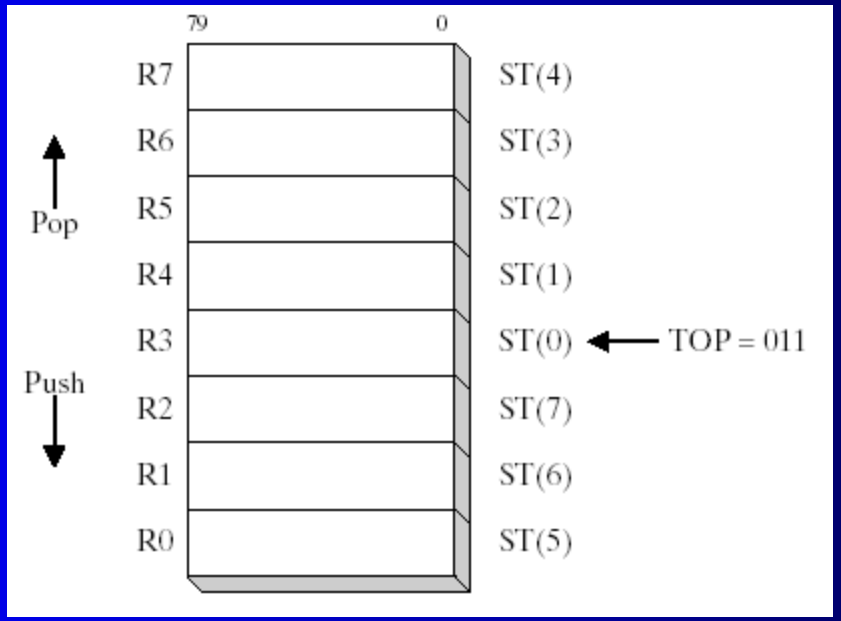
Procesamento de Ponto Flutuante

Unidade ponto-flutuante

- Pilha de registradores FPU
- Arredondamento
- Exceções de ponto-flutuante
- Conjunto de instruções ponto-flutuante
- Instruções aritméticas
- Comparando valores ponto-flutuante
- Lendo e escrevendo valores ponto-flutuante
- Sincronização de exceções
- Aritmética no modo misto
- Mascarando e não-mascarando exceções

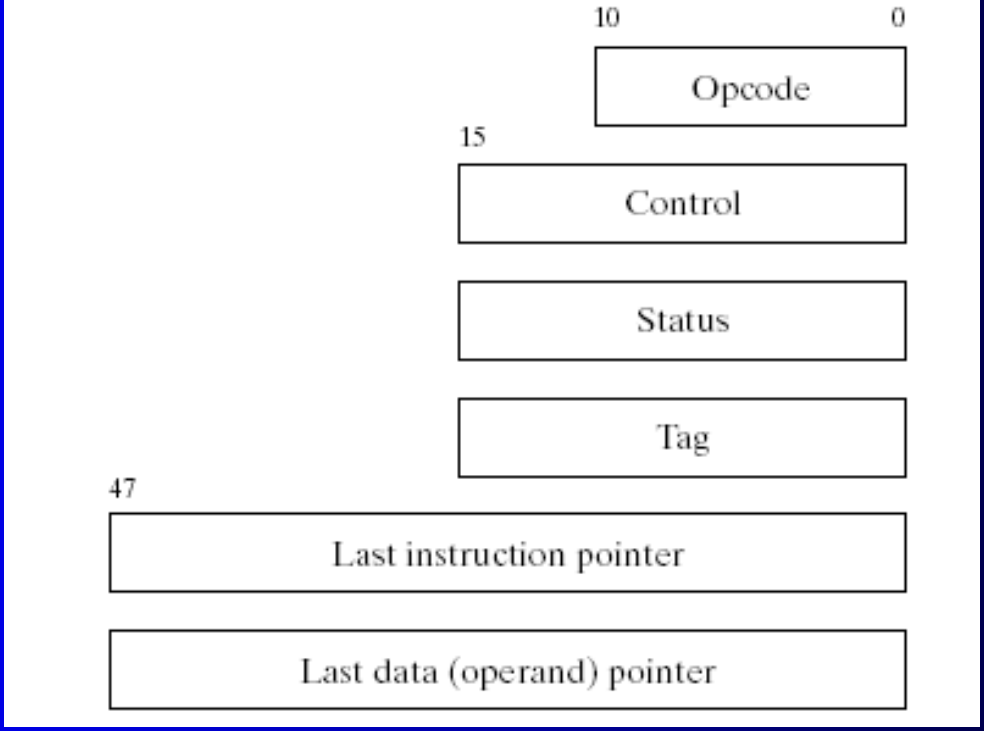
Pilha de registradores FPU

- 8 registradores de dados de 80 bits, endereçáveis individualmente, denominados R0 a R7
- Um campo de 3 bits denominado TOP na palavra de status FPU identifica o número do registrador que é o atual topo da pilha.



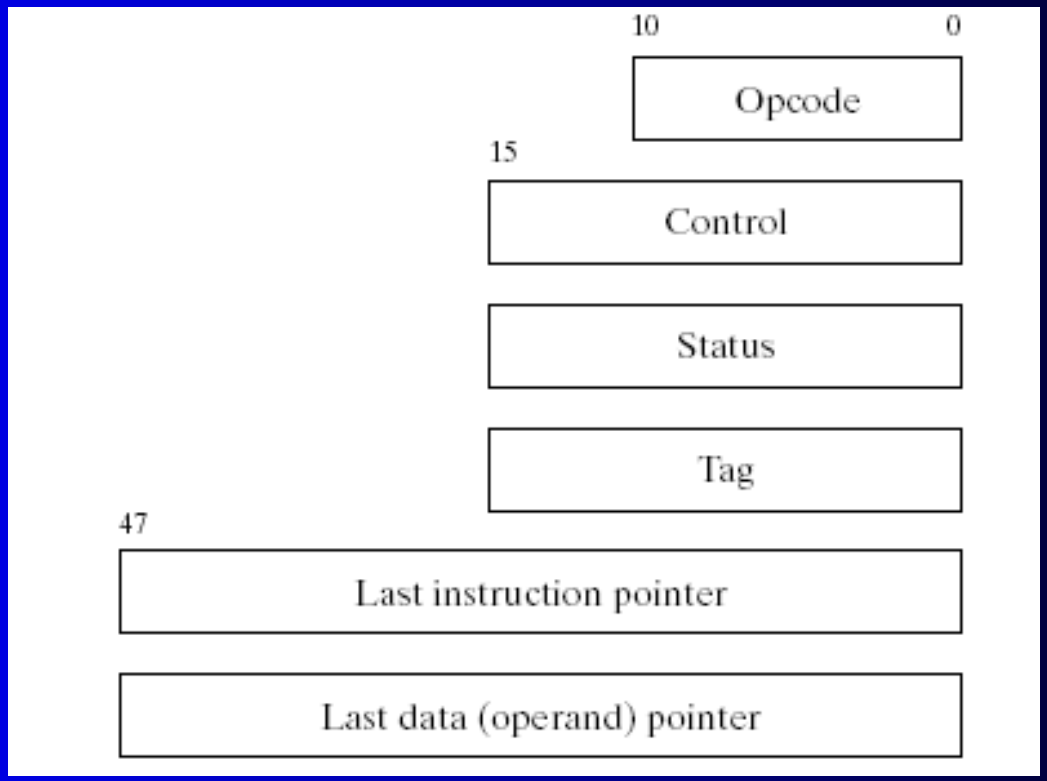
Registradores de propósito especial (1 de 2)

- Opcode : guarda o opcode da última instrução não-controle executada
- Control : controla os métodos de precisão e arredondamento para cálculos
- Status : ponteiro de topo da pilha, códigos de condição, avisos de exceção
- Tag : indica o tipo do conteúdo de cada registrador na pilha



Registradores de propósito especial (2 de 2)

- Last instruction pointer: ponteiro p/ última instrução de “não-controle”
- Last data (operand) pointer: ponteiro do operando da última instrução



Arredondamento

- FPU tenta arredondar o resultado de um cálculo de ponto-flutuante
  - Pode ser impossível devido a limitações de armazenamento
- Exemplo
  - Supor que 3 bits fracionários podem ser guardados e o valor calculado é +1.0111
  - O arredondamento para cima somando .0001 produz 1.100
  - O arredondamento para baixo subtraindo .0001 resulta em 1.011

Exceções de ponto-flutuante

- Seis tipos de condições de exceção:
  - Invalid operation
  - Divide by zero
  - Denormalized operand
  - Numeric overflow
  - Inexact precision
- Cada um tem um bit de máscara (mask bit)
  - Se é igual a 1, quando ocorre, a exceção é manipulada automaticamente pela FPU
  - Se é igual a 0, quando ocorre, um handler de exceção de software é chamado

Conjunto de instruções FPU

- Os mnemônicos de instruções começam com a letra F
- Letras identificam o tipo de dados do operando de memória:
  - B = bcd
  - I = integer
  - no letter: floating point
- Exemplos
  - FLBD load binary coded decimal
  - FISTP store integer and pop stack
  - FMUL multiply floating-point operands

Conjunto de instruções FPU

- Operandos
  - zero, um ou dois
  - Nenhum operando imediato
  - Nenhum registrador de propósito geral (EAX, EBX, ...)
  - Inteiros devem ser carregados da memória para a pilha e convertidos para ponto-flutuante antes de serem usados em cálculos
  - Se uma instrução tem dois operandos, pelo menos um deve ser um registrador da FPU

- OBS: Quanto iniciar o programa que vai utilizar instruções de ponto flutuante, executar a instrução **finit**

código a →

Conjunto de instruções FP

- Tipos de dados

Table 17-11 Intrinsic Data Types.

Type	Usage
QWORD	64-bit integer
TBYTE	80-bit (10-byte) integer
REAL4	32-bit (4-byte) IEEE short real
REAL8	64-bit (8-byte) IEEE long real
REAL10	80-bit (10-byte) IEEE extended real

Carregar valor ponto-flutuante

- FLD
- Copia o operando ponto-flutuante da memória para o topo da pilha FPU, ST(0)

```
FLD m32fp
FLD m64fp
FLD m80fp
FLD ST(i)
```

- Exemplo

```
.data
dblOne    REAL8 234.56
dblTwo    REAL8 10.1
.code
fld  dblOne                    ; ST(0) = dblOne
fld  dblTwo                    ; ST(0) = dblTwo, ST(1) = dblOne
```

Guardar (store) ponto-flutuante

- FST
  - Copia o operando do topo da pilha FPU para a memória
- FSTP
  - Faz o pop da pilha após a cópia

```
FST  m32fp
FST  m64fp
FST  ST(i)
```

código b →

Instruções aritméticas

- Mesmos tipos de operandos que FLD e FST

TABLE 17-12 Basic Floating-Point Arithmetic Instructions.

FCHS	Change sign
FADD	Add source to destination
FSUB	Subtract source from destination
FSUBR	Subtract destination from source
FMUL	Multiply source by destination
FDIV	Divide destination by source
FDIVR	Divide source by destination

Web site Examples

Adição de ponto-flutuante

- FADD
  - Soma fonte ao destino
  - Exemplos

```
FADD[4]
FADD m32fp
FADD m64fp
```

```
fadd
```

Before:	ST(1)	234.56
	ST(0)	10.1
After:	ST(0)	244.66

```
fadd st(1), st(0)
```

Before:	ST(1)	234.56
	ST(0)	10.1
After:	ST(1)	244.66
	ST(0)	10.1

Subtração ponto-flutuante

- FSUB
  - Subtrai fonte do destino.
  - Exemplos:

```
FSUB[5]
FSUB m32fp
FSUB m64fp
FSUB ST(0), ST(i)
FSUB ST(i), ST(0)
```

```
fsub mySingle                    ; ST(0) -= mySingle
fsub array[edi*8]                ; ST(0) -= array[edi*8]
```

Web site Examples

Multiplicação ponto-flutuante

- FMUL
  - Multiplica fonte pelo destino, e guarda o produto no destino

```
FMUL[6]
FMUL m32fp
FMUL m64fp
FMUL ST(0), ST(i)
FMUL ST(i), ST(0)
```

- FDIV
  - Divide o destino pela fonte, e faz o pop da pilha

```
FDIV[7]
FDIV m32fp
FDIV m64fp
FDIV ST(0), ST(i)
FDIV ST(i), ST(0)
```

As versões sem operando de FMUL e FDIV faz o pop da pilha após a multiplicação ou divisão.

Instruções aritméticas

- Instruções aritméticas sem operandos utilizam os dois primeiros valores da pilha – ST(0) e ST(1), armazenando o resultado em ST(0).
- Isso pode ser usado para implementar expressões utilizando a notação PostFiX

- Exemplo:
  - Infix → (A + B) * (C + D)

  - PostFix → A B + C D + *

código c,d →

Comparando valores FP

- Instrução FCOM
- Operandos:

Instruction	Description
FCOM	Compare ST(0) to ST(1)
FCOM *m32fp*	Compare ST(0) to *m32fp*
FCOM *m64fp*	Compare ST(0) to *m64fp*
FCOM ST(*i*)	Compare ST(0) to ST(*i*)

Web site Examples

FCOM

- Códigos de condição usados pela FPU
  - Códigos similares aos flags de CPU

Condition	C3 (Zero Flag)	C2 (Parity Flag)	C0 (Carry Flag)	Conditional Jump to Use
ST(0) > SRC	0	0	0	JA, JNBE
ST(0) < SRC	0	0	1	JB, JNAE
ST(0) = SRC	1	0	0	JE, JZ
Unordered[a]	1	1	1	(*None*)

[a]If an invalid arithmetic operand exception is raised (because of invalid operands) and the exception is masked, C3, C2, and C0 are set according to the row marked *Unordered*.

Desviando após FCOM

- Passos requeridos:
  1. Usar a instrução FNSTSW para mover a palavra de status da FPU em AX.
  2. Usar a instrução SAHF para copiar AH no registrador EFLAGS.
  3. Usar JA, JB, etc para fazer o desvio.

Felizmente (a partir do Pentium P6) , a instrução FCOMI faz os passos 1 e 2 - Exemplo:

```
fcomi ST(0), ST(1)
jnb   Label1
```

Comparação de valores em ponto flutuante

- Lembre-se bem destes slides para evitar entrar na lista de desastres devido a erros de cálculos ☹
- Nem sempre dois valores que achamos iguais são iguais em representação de ponto flutuante.
- Isso ocorre devido a diferenças de arredondamento.
- Assim, devemos sempre permitir um fator de tolerância Epsilon na comparação de dois valores.

Comparação de valores em ponto flutuante

- Calcular o valor absoluto da diferença entre dois valores ponto-flutuante

```
.data
epsilon REAL8 1.0E-12          ; difference value
val2 REAL8 0.0                 ; value to compare
val3 REAL8 1.001E-13           ; considered equal to val2

.code
; if( val2 == val3 ), display "Values are equal".
    fld epsilon
    fld val2
    fsub val3
    fabs
    fcomi ST(0),ST(1)
    ja skip
    mWrite <"Values are equal",0dh,0ah>
skip:
```

Comparação de valores em ponto flutuante

- Exemplo: (sqrt(2.0) * sqrt(2.0)) – 2.0 = 0.0 ?

Comparação de valores em ponto flutuante

- Exemplo: (sqrt(2.0) * sqrt(2.0)) – 2.0 = 0.0 ?

Não, a valor calculado é:

+4.4408921E-016

código e, f →

Entrada/saída ponto-flutuante

- Procedimentos Irvine32 library
  - ReadFloat
    - Lê um valor FP do teclado e coloca na pilha FPU
  - WriteFloat
    - Escreve valor de ST(0) na tela em formato exponencial
  - ShowFPUStack
    - Mostra o conteúdo da pilha FPU

Mascarando e não-mascarando exceções

- Exceções mascaradas (por default):
  - Divide by zero gera infinito, sem parar o programa
- Exceção não-mascarada:
  - O processador executa um handler de exceção apropriado
  - Não-mascarar a exceção de divide by zero zerando o bit 2 do registrador control:

```
.data
ctrlWord WORD ?
.code
fstcw ctrlWord                      ; get the control word
and ctrlWord,1111111111111011b      ; unmask divide by zero
fldcw ctrlWord                      ; load it back into FPU
```

Sincronização de exceções

- A CPU e a FPU podem executar instruções concorrentemente
  - Se ocorre uma exceção não-mascarada, a instrução FPU corrente é interrompida e a FPU sinaliza uma exceção
  - Mas a CPU não checa pelas exceções de FPU pendentes. Ela deve usar um valor de memória que a instrução de FPU interrompida supostamente teria escrito.
  - Exemplo:

```
.data
intVal DWORD 25
.code
fild intVal          ; load integer into ST(0)
inc  intVal          ; increment the integer
```

Sincronização de exceções

- (continuação)
- Para segurança, inserir uma instrução fwait, que diz à CPU esperar pelo handler de exceção de FPU terminar:

```
.data
intVal DWORD 25
.code
fild intVal         ; load integer into ST(0)
fwait               ; wait for pending exceptions
inc intVal          ; increment the integer
```

código g →

Aritmética no modo misto

- Combinando inteiros e reais.
  - Instruções de aritmética inteira tais como ADD e MUL não podem manipular reais
  - FPU tem instruções que convertem inteiros para reais e carregam os valores na pilha de ponto-flutuante.
- Exemplo: Z = N + X

```
.data
N SDWORD 20
X REAL8 3.5
Z REAL8 ?
.code
fild N                  ; load integer into ST(0)
fwait                   ; wait for exceptions
fadd X                  ; add mem to ST(0)
fstp Z                  ; store ST(0) to mem
```

código h →

Instruções de Ponto Flutuante

Instrução	descrição	instrução	descrição
F2XM1	Calculates 2^x-1	FABS	Absolute value
FADD	Add real	FADDP	Add real and pop
FBLD	Load bcd	FBSTP	Store bcd and pop
FCHS	Change sign	F(N)CLEX	Clear exceptions
FCOM	Compare real	FCOMP	Compare real and pop
FCOMPP	Compare real and pop twice	FDECSTP	Decrement stack
FDIVP	Divide real and pop	FDIV	Divide real
FDIVRP	Divide real reversed and pop	FDIVR	Divide real reversed
FFREE	Free register	FIADD	Integer add
FICOM	Compare integer	FICOMP	Compare integer and pop
FIDIV	Integer divide	FIDIVR	Integer divide reversed
FILD	Load integer	FIMUL	Integer multiply
FINCSTP	Increment stack	F(N)INIT	Initialize coprocessor
FIST	Store integer	FISTP	Store integer and pop
FISUB	Integer subtract	FISUBR	Integer subtract reversed

Web site Examples

Instruções de Ponto Flutuante

FLD	Load real	FLD1	Load value of 1.0
FLDCW	Load control word	FLDENV	Load environment state
FLDL2E	Load value of log2(e)	FLDL2T	Load value of log2(10)
FLDLG2	Load value of log10(2)	FLDLN2	Load value of loge(2)
FLDPI	Load value of pi	FLDZ	Load value of 0.0
FMUL	Real multiply	FMULP	Real multiply and pop
FNOP	No operation	FPATAN	Partial arctangent
FPREM	Partial remainder	FPTAN	Partial tangent
FRNDINT	Round to integer	FRSTOR	Restore saved state
F(N)SAVE	Save coprocessor state	FSCALE	Scale with powers of 2
FSQRT	Square root	FST	Store real
F(N)STCW	Store control word	F(N)STENV	Store environment state
FSTP	Store real and pop	F(N)STSW	Store status word
FSUB	Real subtract	FSUBP	Real subtract and pop
FSUBPR	Real subtract reversed and pop	FSUBR	Real subtract reversed
FTST	Test for zero	FWAIT	CPU Wait (until FPU ready)
FXAM	Examine	FXCH	Exchange registers
FXTRACT	Extract exponent and mantissa	FYL2X	Calculates y. log2(x)
FYL2XP1	Calculates y. log2 (x+1)		

Instruções de Ponto Flutuante

Instrução	Descrição
FCOS	Cosine
FLDENVW/D	Load environment state
FPREM1	Partial reminder (IEEE)
FSIN	Sine
FSINCOS	Sine and cosine
FUCOM	Unordered compare
FUCOMP	Unordered compare and pop
FUCOMPP	Unordered compare and pop twice

Web site Examples

Exemplos de Código

Exemplos a-g, mostrados em aula → disponíveis no Moodle

Fim