


Sistemas de numeração
Representação flutuante e dupla
CISC e RISC

1

Citação importante!

- **Existem 10 tipos de pessoas no mundo: As que entendem binário e as que não entendem.**

2

2^0	1		Para saber todos os valores
2^1	2		
2^2	4		
2^3	8		
2^4	16		
2^5	32		
2^6	64		
2^7	128		
2^8	256		
2^9	512		
2^{10}	1024=1K		
2^{16}	64K		
2^{20}	1M		
2^{24}	16M		
2^{30}	1G		
2^{32}	4G		
2^{36}	64G		
2^{40}	1T		

3

O básico

- Humanos: normalmente usam o sistema decimal para cálculos (embora existam outros sistemas)
- Computadores: normalmente usam o sistema binário para cálculos

è Assim, é necessária uma conversão

- Além disso, os sistemas de computador usam outras representações, como octal ou hexadecimal para a representação mais compacta de números binários maiores

è Portanto, é importante entender alguns fundamentos matemáticos e relações entre os diferentes sistemas de numeração

4

Sistemas Numéricos

- Sistema hexadecimal: normalmente usamos as letras de A a F para representar os dígitos com valores de 10 a 15
- Sistema binário: sistema mais importante dentro de um computador
- Sistemas octal e hexadecimal: muito simples de converter para o sistema binário, mais fácil de ler.

Base (b)	sistema numérico	Alfabeto
2	Sistema binário	0,1
8	Sistema octal	0,1,2,3,4,5,6,7
10	Sistema decimal	0,1,2,3,4,5,6,7,8,9
16	Sistema hexadecimal	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

5

Diferentes sistemas de numeração (4 bits)

Binário	Decimal (sem sinal)	Sinal + Módulo	2 complementos	Decimais hexadecimais
0000	0	0	0	0
0001	1	1	1	
0010	2	2	2	1 2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	0	-8	8
1001		-1	-7	9
1010	9 10	-2	-6	A
1011	11	-3	-5	B
1100	12	-4	-4	C
1101	13	-5	-3	D
1110	14	-6	-2	E
1111	15	-7	-1	F

6

Conversão entre bases

binário para decimal

Soma as potências de 2 de todos os dígitos com 1

7 6 5 4 3 2 1 0

$$01100010 = 2^6 + 2^5 + 2^1 = 64 + 32 + 2 = 98$$

7 6 5 4 3 2 1 0

$$01111101 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 64 + 32 + 16 + 8 + 4 + 1 = 125$$

Decimal para binário

$$40 = 32 + 8 = 00101000$$

$$60 = 64 - 4 = 63 - 3 = 00111111 - 00000011 = 00111100$$

$$120 = 128 - 8 = 127 - 7 = 01111111 - 00000111 = 01111000$$

7

Conversão entre bases

Binário para

hexadecimal 1º Juntar grupos de 4 bits da direita para a

esquerda 2º Converter cada grupo de 4 bits em um único símbolo

hexadecimal 11101110010100010111010110011111

EE 5 1 7 5 9 F

Hex para binário

Converter cada símbolo hexadecimal em 4 bits

Binário	Decimais hexadecimais
0000	0
0001	
0010	1 2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

8

24 = 16 + 4 = 4 dígitos binários @ 1 dígito hexadecimal

dual

0110100.110101



0011 0100 1101 0100

hexadecimais 3 4 . D 4

Preencha os zeros que faltam para obter grupos completos de 4 dígitos.

9

números negativos

- Valor absoluto mais sinal (V+S)
- Complemento de uns
- Complemento de dois

10

Representação com valor absoluto mais sinal (V+S)

- Um dígito representa o sinal, normalmente o MSB
 - MSB = bit mais significativo
- O bit mais à esquerda representa o sinal de um número (por convenção)
 - MSB = 0 \Rightarrow número positivo
 - MSB = 1 \Rightarrow número negativo
- Exemplo:

• 0001 0010 •	= +18
1001 0010	= -18
- Desvantagens:
 - Manipulação separada dos sinais durante a adição e subtração
 - Existem duas representações do número 0
 - Um com sinal positivo e outro com sinal negativo (+0 e -0)

11

complemento de uns

- Inverte todos os bits simples de um número binário para obter o número com um sinal invertido.
- Isso é chamado de complemento de um
 - n é o número de dígitos, por exemplo, $n=4 \Rightarrow$ números de 4 bits
- Exemplo:

410 = 01002	\Rightarrow -410 = 10110c
-410 = (24 - 1) - 4 = 1110 = 10112	
- Novamente, números negativos têm o MSB = 1
- Vantagem (comparado ao valor absoluto mais sinal)
 - Nenhum manuseio separado do MSB durante a adição ou subtração
- Desvantagem
 - Ainda duas representações de zero (0000 e 1111 para números de 4 bits)

12

complemento de dois

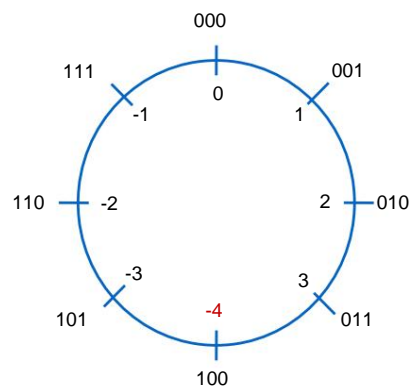
- Evite a desvantagem adicionando 1 após aplicar o complemento de uns:
- Isso resulta no complemento de dois:
- Apenas uma representação do zero!

$0 \dots 0$ $\neq 1 \dots 10c$
 $\neq 0 \dots 0tc$

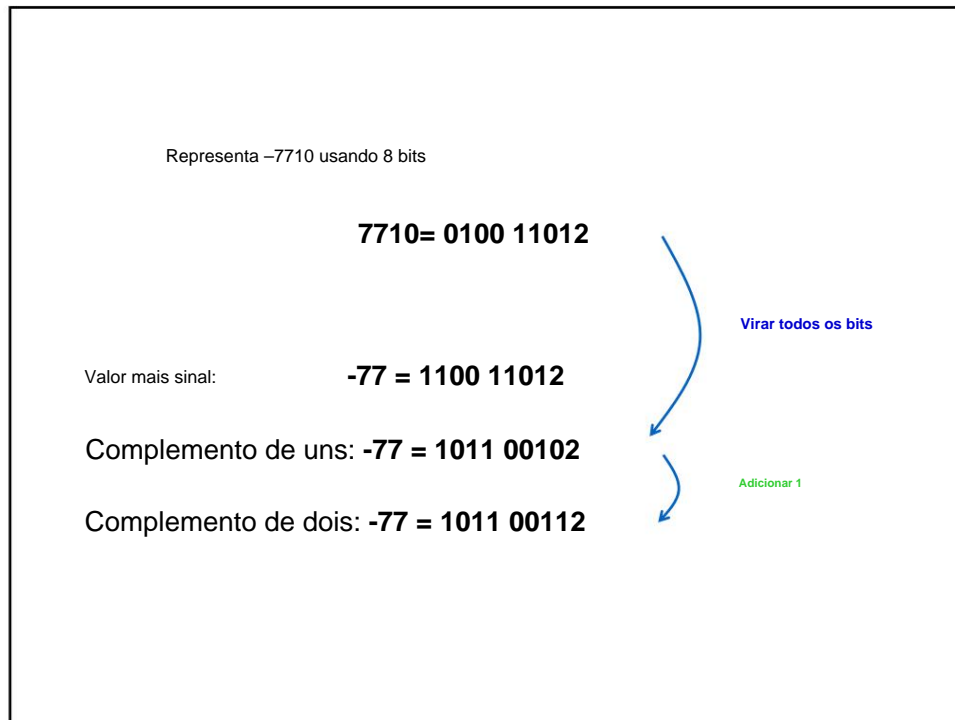
13

complemento de dois

- Desvantagem: •
Intervalo assimétrico de números que podem ser representados
 - O número mais baixo tem um valor absoluto maior (em 1) do que o número mais alto
- Exemplo: números de complemento de dois de 3 bits
- Novamente, números negativos têm o MSB = 1



14

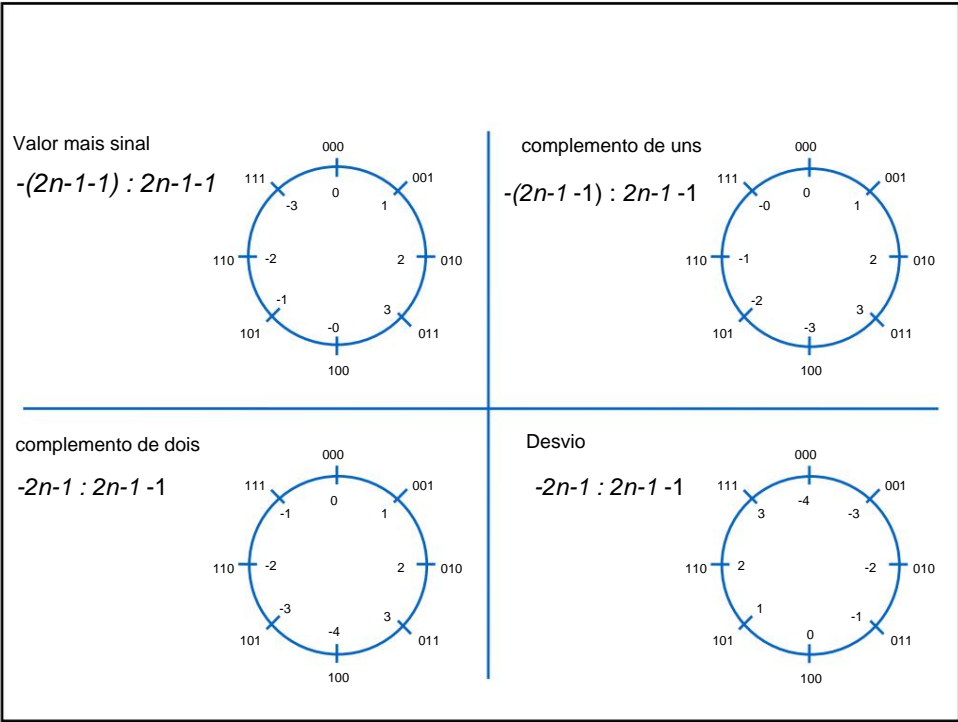


15

Deslocar representação binária/excesso/tendenciosa

- Comumente usado para a representação de expoentes de números de ponto flutuante (mas também, por exemplo, no processamento de sinais, pois os conversores são unipolares, ou seja, eles não podem lidar com valores negativos).
- Essa representação de um expoente também é chamada de **característica**.
- Todo o intervalo de números é deslocado pela adição de um valor constante (offset/excess/bias) para que o menor número (maior valor negativo) obtenha a representação **0...0**.
- Assumindo n dígitos: **Offset** = 2^{n-1}
 - Exemplo: $n=8$ / Offset 128
- O intervalo de números é **assimétrico**.

16



17

Tamanho (pedaço)	nomes típicos	Sinal	Intervalo numérico (usando complemento de dois)	
			min	máximo
8	char, octeto, byte, moderno: <code>int8_t</code> ou <code>uint8_t</code>	assinado	$\bar{y}128$	127
		não assinado	0	255
16	Word, Short/short, Inteiro, moderno: <code>int16_t</code> ou <code>uint16_t</code>	assinado	$\bar{y}32.768$	32.767
		não assinado	0	65.535
32	DWord/Double Word, int, long (Windows em 16/32/64 bits sistemas; Unix/Linux em 16/32 bits sistemas), moderno: <code>int32_t</code> ou <code>uint32_t</code>	assinado	$\bar{y}2.147.483.648$	2.147.483.647
		não assinado	0	4.294.967.295
64	Int64, QWord/Quadword, longo long, Long/long (Unix/Linux em sistemas de 64 bits), moderno: <code>int64_t</code> ou <code>uint64_t</code>	assinado	$\bar{y}9.223.372.036.854.775.808$	9.223.372.036.854.775.807
		não assinado	0	18.446.744.073.709.551.615
128	Int128, Octaword, Duplo Quadword	assinado	$\bar{y} \bar{y}1,70141-1038$	$\bar{y} 1,70141-1038$
		não assinado	0	$\bar{y} 3,40282-1038$

18

Tipo de ponto flutuante	Requisito de memória	Faixa
Flutuador	4 bytes	$\pm 3,40282347\text{E}+38\text{F}$, ou seja, 6-7 dígitos significativos
Dobro	8 bytes	$\pm 1,79769313486231570\text{E}+308$ ou seja, 15-16 dígitos significativos

19

Representação de números flutuantes precisão simples (32 bits)

0: Positivo

1: Negativo

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Expoente (8 bits)
Excesso de 127Parte decimal
23 bits

sinal (1, parte decimal)*2(expoente-127)

1 2 3 4 5 6 7 8 9 23
XXXXXXXXXXXXXXXXXXXXXXXXXXXX

$$\frac{1}{2^1} = 0,5$$

$$\frac{1}{2^2} = 0,25$$

$$\frac{1}{2^3} = 0,125$$

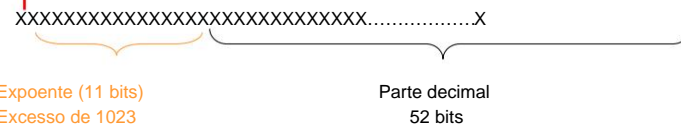
$$\frac{1}{2^{23}} = 0,000\dots$$

20

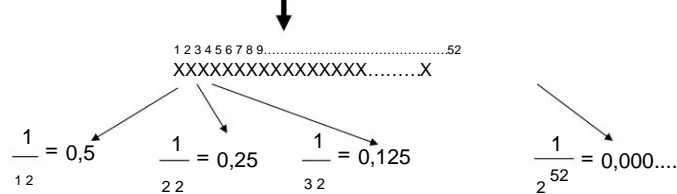
Representação de números flutuantes precisão dupla (64 bits)

0: Positivo

1: Negativo



sinal (1, parte decimal) * 2^(expoente-1023)



21

Representação de números flutuantes exercícios práticos

Regras:

1. Coloque o número no formato de sinal (1, parte decimal) * 2^(expoente)

Se o número for maior ou igual a 2

Enquanto o número for maior ou igual a 2

Divida o número por 2

(o expoente é o número de divisões)

Else se o número for menor que 1

Enquanto o número é menor que 1

multiplique o número por 2 (o

expoente é o número negativo de multiplicações)

Outro

(o expoente é 0)

2. Coloque o número no **sinal** de formato (1, parte decimal) * 2^(expoente-127 ou-1023)

3. Encontre o bit para sinal, os bits para parte decimal e os bits para expoente

22

Representação de números flutuantes

exercícios práticos

Representar em precisão simples 3,5

Como o número é maior ou igual a 2, devemos dividir o número até obter um número menor que 2.

$$3,5/2=1,75$$

$$3,5= 1,75 \cdot 2^1$$

$$3,5=1,75 \cdot 2^{(128-127)}$$

Sinal 0

Parte decimal 0,75=0,5+0,25. Sequência de bits: 110000000000000000000000

Expoente = 128. Sequência de bits: 10000000

Resultado final: 0 10000000 110000000000000000000000

Resultado final em hexadecimal: 4060000H

23

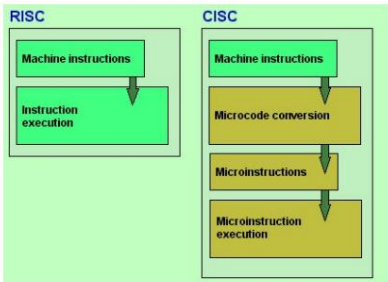
exercícios

Por favor, represente nos formatos **Float** e **Double** os seguintes números decimais. Identifique claramente o bit de sinal, o expoente e a parte decimal.

- a) 2
- b) -2
- c) 4
- d) 6,5
- e) 10

24

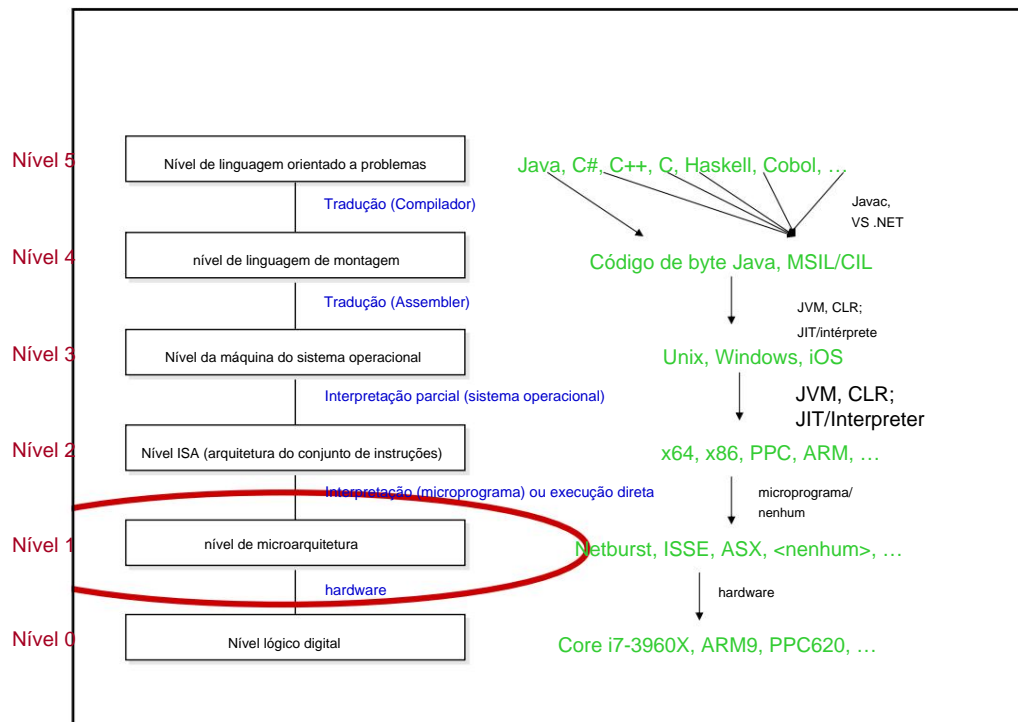
CISC (computador de conjunto de instruções complexo)
vs RISC (computador de conjunto de instruções reduzido).



25

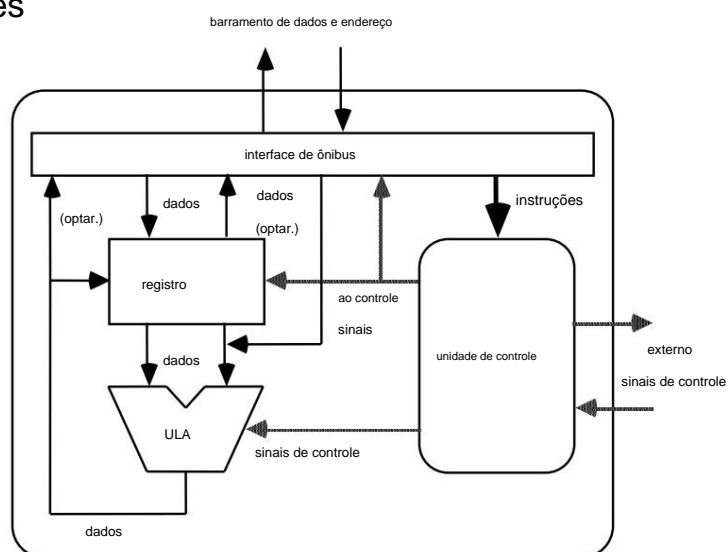
RISC	CISC
1. RISC significa Conjunto de Instruções Reduzido Computador.	1. CISC significa Complex Instruction Set Computer.
2. Os processadores RISC têm instruções simples que levam cerca de um ciclo de clock.	2. O processador CSIC possui instruções complexas que consomem vários clocks para execução.
3. O desempenho é otimizado com mais foco no 3. O desempenho é otimizado com mais foco no software hardware.	
4. Não possui unidade de memória e usa uma unidade separada hardware complexo para implementar instruções.	4. Possui uma unidade de memória para implementar instruções.
5. O conjunto de instruções é reduzido, ou seja, tem apenas 5. O conjunto de instruções possui uma variedade de poucas instruções diferentes no conjunto de instruções. Muitas das instruções complexas são muito primitivas.	5. O conjunto de instruções possui uma variedade de poucas instruções diferentes no conjunto de instruções. Muitas das instruções complexas são muito primitivas.
6. Modos de endereçamento complexos são sintetizados 6. CISC já suporta endereçamento complexo usando o software.	6. CISC já suporta endereçamento complexo usando o software.
7. Vários conjuntos de registradores estão presentes 7. Possui apenas um único conjunto de registradores	7. Possui apenas um único conjunto de registradores
8. O tempo de execução é muito menor 8. O tempo de execução é muito alto	8. O tempo de execução é muito alto
9. A decodificação das instruções é simples.	9. A decodificação de instruções é complexa
10. Não requer memória externa para cálculos 11. Os	10. Requer memória externa para cálculos
microprocessadores RISC mais comuns são Alpha, ARC, ARM, AVR, MIPS, PA-RISC, PIC, Power Architecture e SPARC.	11. Exemplos de processadores CISC são System/360, VAX, PDP-11, família Motorola 68000, CPUs AMD e Intel x86.
12. A arquitetura RISC é usada em aplicativos de ponta, como processamento de vídeo, telecomunicações e processamento de imagem.	12. A arquitetura CISC é usada em aplicativos de baixo custo, como sistemas de segurança, automação residencial, etc.

26



27

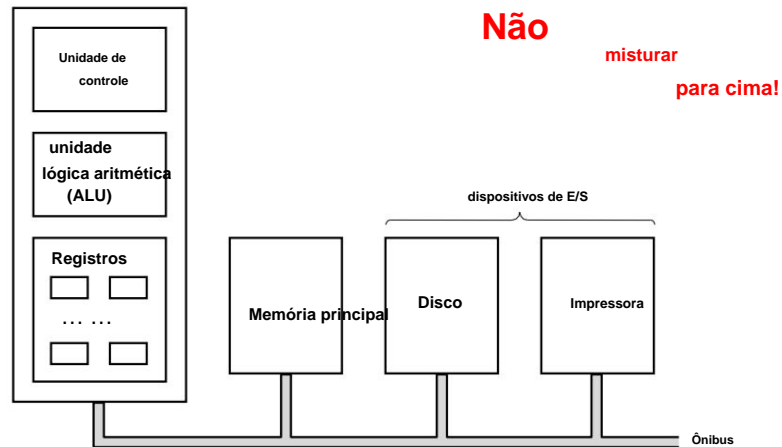
Arquitetura básica de um microprocessador simples



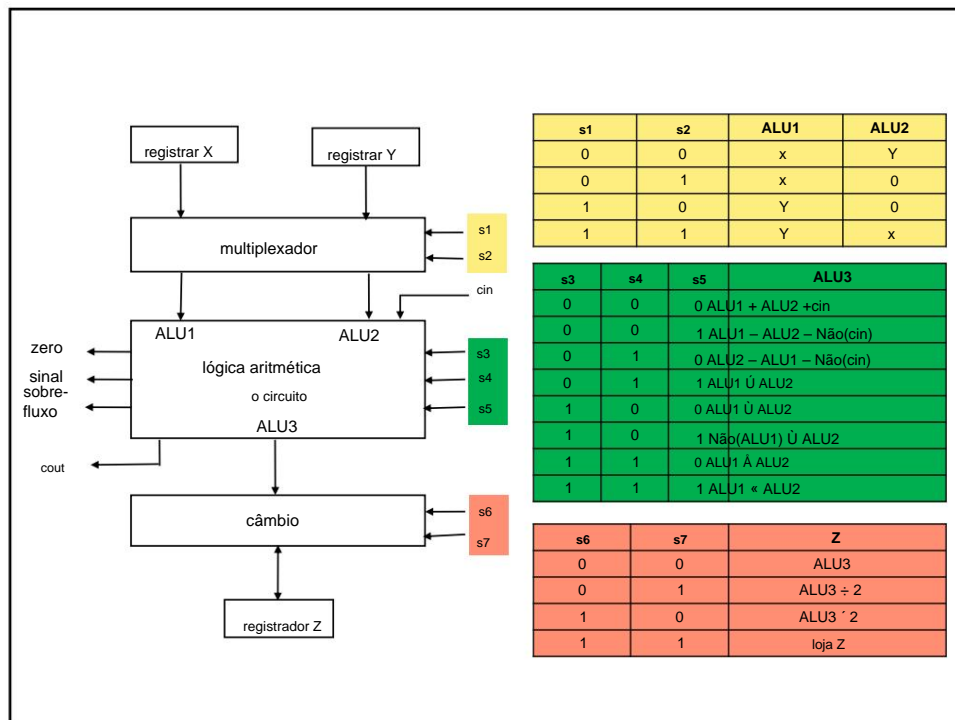
28

Arquitetura de processador único

Unidade de processamento central (CPU)

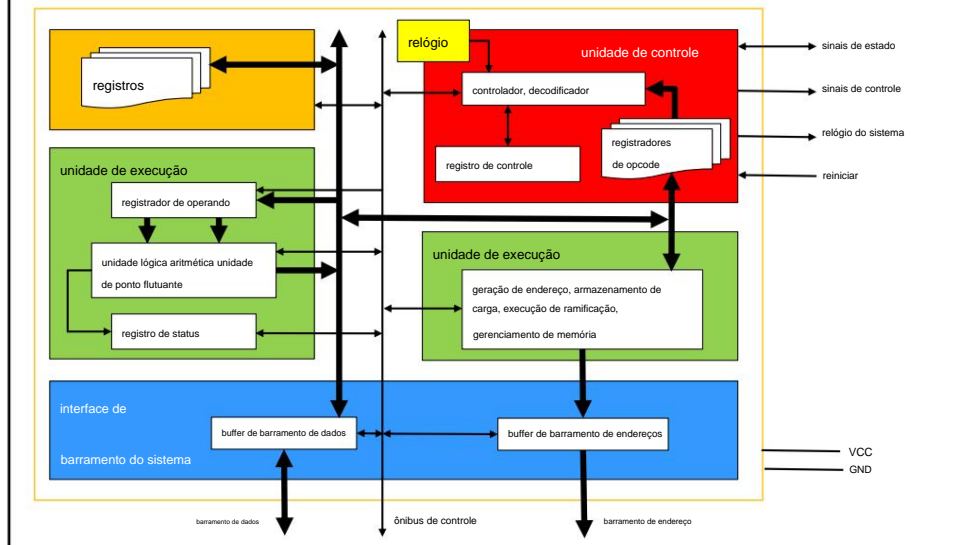


29



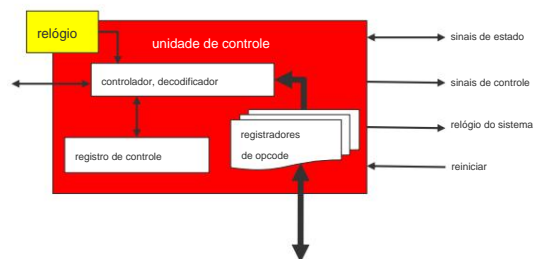
30

Arquitetura interna de um microprocessador simples e simplificado



31

Unidade de controle



32

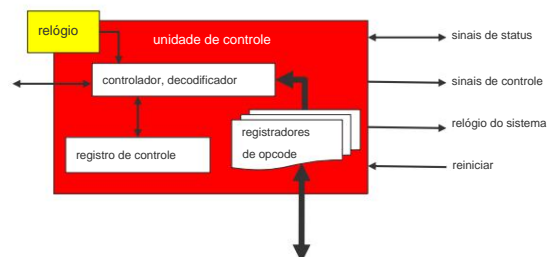
Unidade de controle

- A unidade de controle controla todos os componentes
- O **relógio** gera o relógio do sistema para distribuição a todos os componentes
- Os **registradores Opcode** contêm a parte da instrução que especifica a operação atualmente executada a ser realizada (e talvez alguns opcodes adicionais)
- O **decodificador** (geralmente microprogramável) gera todo o controle sinais para os componentes e usa sinais de status e opcode como entrada
- O **registro de controle** armazena o status atual da unidade de controle

33

Cronometragem / sincronização

- **Circuito sequencial síncrono**
 - Normalmente, as CPUs usam lógica dinâmica (com clock)
 - O estado é armazenado em capacitâncias de porta
 - A lógica estática usa flip-flops
 - Velocidade mínima de clock necessária
- Caso contrário, os bits armazenados são perdidos devido a vazamento antes do próximo ciclo de clock
- Rede de distribuição de relógio complexa no chip necessária



34

Unidade de controle micro programável

- O processador armazena um **microprograma** para cada instrução
 - Microprograma: sequência de microinstruções •
- Usuários normais não podem alterar o microprograma de um processador
- No entanto, os fabricantes podem atualizar o microprograma
- Processadores RISC puros normalmente não usam microprogramas, mas um circuito sequencial fixo.

35

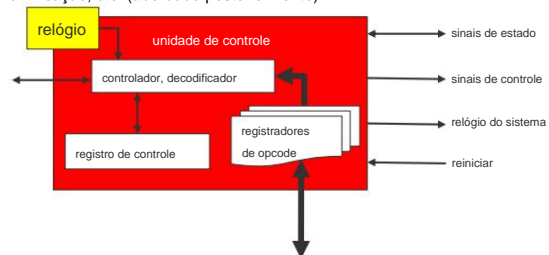
Fases da execução da instrução

- **Busca de instrução**
 - Carregar a próxima instrução no registrador opcode
- **Decodificação de instrução**
 - Obtenha o endereço inicial do microprograma que representa a instrução
- **Execução**
 - O microprograma controla a execução da instrução enviando os sinais apropriados para os outros componentes e avaliando os sinais retornados

36

registrador de opcode

- O registrador opcode consiste em vários registradores porque
 - instruções diferentes podem ter tamanhos diferentes (1 byte, 2 bytes, 3 bytes...)
 - a pré-busca do opcode pode acelerar a execução do programa • durante a decodificação da instrução atual, as seguintes instruções podem ser pré-buscadas • isso suporta pipelining, previsão de ramificação, etc. (abordado posteriormente)



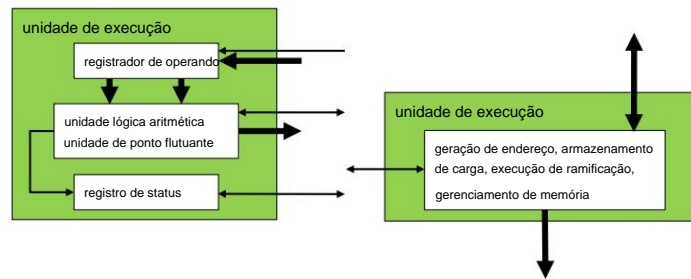
37

Registro de controle

- O registro de controle armazena o estado atual da unidade de controle. • Isso influencia, por exemplo, a decodificação da instrução, o modo de operação.
- O significado dos bits depende do processador.
- Exemplos:
 - Bit de habilitação de interrupção
 - determina se o processador reage a interrupções
 - Extensões de máquinas virtuais permitem
 - ativar a virtualização assistida por hardware em CPUs x86
 - Prevenção de instrução do modo de usuário
 - se definido, certas instruções não podem ser executadas no nível do usuário

38

unidade de execução



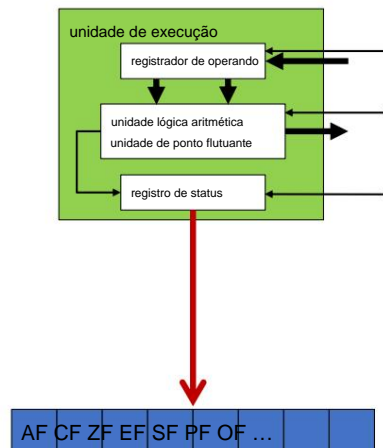
39

unidade de execução

- A **unidade de execução** executa todas as operações lógicas e aritméticas controladas pela unidade de controle.
- Exemplos:
 - Operações aritméticas inteiras e flutuantes
 - Operações lógicas, deslocamento, comparações
 - Todas as operações relacionadas a endereços
 - Operações especulativas (abordadas posteriormente)
 - Gerenciamento de memória complexo, proteção de memória
 - ...
- O **registro de status** informa a unidade de controle sobre o estado do processador após uma operação
 - Exemplos: carry, overflow, zero, sign
- **Registradores de operandos, acumuladores** etc.: registradores adicionais para resultados temporários, operadores buscados etc.

40

Registro de status (registro de sinalizador, código de condição
Registrar CCR)



41

Bandeiras

- **Overflow Flag (OF)** ̃ Indica o estouro de um bit de alta ordem (bit mais à esquerda) de dados após uma operação aritmética com sinal.
- **Sinalizador de Direção (DF)** ̃ Determina a direção esquerda ou direita para mover ou comparando dados de string. Quando o valor DF é 0, a operação de string ocorre da esquerda para a direita e quando o valor é definido como 1, a operação de string ocorre da direita para a esquerda.
- **Sinalizador de interrupção (IF)** ̃ Determina se as interrupções externas, como entrada de teclado, etc., devem ser ignoradas ou processadas. Desabilita a interrupção externa quando o valor é 0 e habilita as interrupções quando definido como 1.
- **Trap Flag (TF)** ̃ Permite configurar a operação do processador no modo single step. O programa DEBUG que usamos define o sinalizador de trap, para que possamos percorrer a execução uma instrução por vez.

42

Bandeiras

- **Sign Flag (SF)** y Mostra o sinal do resultado de uma operação aritmética. Esse sinalizador é definido de acordo com o sinal de um item de dados após a operação aritmética. O sinal é indicado pela ordem superior do bit mais à esquerda. Um resultado positivo limpa o valor de SF para 0 e um resultado negativo o define como 1.
- **Zero Flag (ZF)** y Indica o resultado de uma operação aritmética ou de comparação. Um resultado diferente de zero limpa o sinalizador zero para 0 e um resultado zero o define como 1.
- **Auxiliary Carry Flag (AF)** y Contém o carry do bit 3 para o bit 4 após um operação aritmética; usado para aritmética especializada. O AF é definido quando uma operação aritmética de 1 byte causa um transporte do bit 3 para o bit 4.
- **Parity Flag (PF)** y Indica o número total de 1-bits no resultado obtido de uma operação aritmética. Um número par de bits 1 limpa o sinalizador de paridade para 0 e um número ímpar de bits 1 define o sinalizador de paridade para 1.
- **Carry Flag (CF)** y Contém o carry de 0 ou 1 de um bit de alta ordem (mais à esquerda) após uma operação aritmética. Ele também armazena o conteúdo do último bit de uma operação *de deslocamento* ou *rotação*.

43

Palavra de status do programa (PSW)

- Registro de status mais registro de controle determinam o **estado atual de um processador**
 - Resultado de uma operação
 - Nível de privilégio
 - ...
- Juntamente com o **contador de programa** (endereço da instrução atual ou seguinte), esses registradores determinam o estado do processador em uma determinada instrução de um programa (ou processo, tarefa, ...).
- O **PSW** combina os registradores e contador de programa para uma manipulação mais simples.
 - Empurrado para empilhar antes da mudança de contexto (por exemplo, mudar para outro processo)
 - Puxado da pilha para continuar a execução de um processo interrompido

44

Operações típicas (simples) de uma ALU

- Aritmética •
 - Adição com/sem transporte •
 - Subtração com/sem transporte •
 - Incremento/decremento
 - Multiplicação com/sem sinal •
 - Divisão com/sem sinal •
 - Complemento de dois
- Lógico
 - NÃO
 - E
 - OU
 - XOR
- Deslocamento e rotação
 - Deslocar para a esquerda
 - Deslocar para a direita • Girar para a direita sem carregar
 - Girar para a direita com carregar •
 - Girar para a esquerda sem carregar •
 - Girar para a esquerda com carregar
- Memória
 - Transferência
 - Carregar, armazenar

45

Computador de Conjunto de Instruções Complexas (CISC)

- Razões para o CISC •
 - Execução de instruções complexas mais rapidamente do que a execução de programas equivalentes com a mesma função
 - Microprogramação permite instruções mais complexas • Instruções mais complexas levam a programas mais curtos, portanto, carregamento mais rápido (intervalo na taxa de transferência entre a CPU interna e a memória principal da CPU) • Quanto maior, melhor – mais instruções soam mais poderosas...é marketing! • Suporte direto de construções de programação de linguagens superiores usando instruções mais complexas (por exemplo, comparação de strings) • Suporte de compiladores poderosos especializados • Compatibilidade (podemos fazer tudo como antes mais xyz) • Suporte de aplicativos de finalidade especial (por exemplo, operações de matriz)
- mais transistores/chip, linguagens de programação mais avançadas e aplicações para fins especiais favorecem instruções “complexas”

46

Computador de Conjunto de Instruções Complexas (CISC)

• Razões contra CISC

- Memórias principais muito mais rápidas (argumento dos anos 80, hoje novamente um problema) e o uso de memória cache para acelerar a execução do programa
- Microprogramas são cada vez mais complexos (então onde está a diferença entre programação e micro programação...)
- Substituição de instruções complexas por várias mais simples (muito mais rápidas) instruções
- Ciclos de desenvolvimento mais longos
- Unidades de controle muito complexas
- Grandes microprogramas com (potencialmente com erros)
- Os programas reais usam frequentemente apenas uma pequena fração do grande conjunto de instruções!

47

As 10 instruções mais usadas no SPECint92 para Intel x86

Instrução	Porcentagem [%]
carregar	22
ramificação condicional	20
comparar	16
loja	12
adicionar	8
e	6
sub	5
mover registrar-registrar	4
chamada	1
retornar	1
Total	95

48

Limitações das arquiteturas CISC

- Uso de instruções (regra 80/20)
 - Apenas 20% das instruções usadas com frequência
 - Muitas instruções poderosas (raramente usadas) •Formato(s) de instrução complexo(s)
 - Microprogramação
- Problema crítico: número de ciclos por instrução (CPI)
 - Muitas arquiteturas CISC clássicas têm $CPI \gg 2$
 - Motorola MC68030: $CPI = 4-6$
 - Intel 80386: $CPI = 4-5$
 - **MAS:** código otimizado para Pentium/Itanium/... – típico $CPI \approx 1$
 - Processadores superescalares, por exemplo, emitir 4 instruções em paralelo poderiam, teoricamente, cair para 0,25, mas: ponto flutuante, SIMD, previsões incorretas de ramificação, latência de memória ...

49

Computador com conjunto de instruções reduzido (RISC)

- O conjunto de instruções consiste em
 - algumas **instruções absolutamente necessárias** (≈ 128) e
 - **formatos de instrução** (≈ 4) com um •**comprimento de instrução fixo** de 32 bits e apenas alguns
 - **modos de endereçamento** (≈ 4).
- Isso permite uma implementação muito mais simples da unidade de controle e economiza espaço no chip para unidades adicionais.
- Muitos registradores de uso geral, pelo menos 32, são necessários.
- O acesso à memória só é possível por meio de instruções especiais de carregamento e armazenamento.

50

Computador com conjunto de instruções reduzido (RISC)

- O acesso à memória é feito apenas por meio de operações de carregamento e armazenamento.
- Todas as outras instruções funcionam apenas nos registradores da CPU, por exemplo, operações aritméticas carregam operandos de registradores e armazenam resultados apenas em registradores.
- Este princípio básico é chamado
 - registrar/registrar arquitetura ou
 - carregar/armazenar arquitetura e é típico para muitos computadores RISC (originais).

51

RISC

- Se possível, todas as instruções devem ser implementadas de forma que terminem em um único ciclo do processador.
- Consequência: processadores RISC puros não usam microprogramação
 - Os processadores RISC introduziram mecanismos aprimorados de pipelining (hoje, muitos processadores usam pipelining para as microinstruções, por exemplo, Pentium 4 e superior).
- Além disso, os primeiros processadores RISC tinham um pipeline controlado por software (compiladores inseriam NOPs de atraso, introduziam saltos atrasados etc.) em vez de hardware especial.
- Aparte
 - Processadores de PC como o Pentium 4 (e superior) usam microprogramação, a microarquitetura interna (netburst) é bastante RISC, o ISA é CISC.

52

RISC

- Razões para

- Implementação de chip único (sim, hoje "tudo" cabe em um único chip)
- Ciclos de desenvolvimento mais curtos
- Taxas de clock mais altas, pipelining
- Reutilização de espaço de chip salvo para, por exemplo, cache

- Razões contra

- Gargalo na interface de memória, hoje novamente a memória principal é muito mais lenta comparada aos registradores/cache internos
- O espaço em um chip não é mais tão crítico

53

CISC / RISC

- RISC puro prefere a arquitetura Harvard

- Memória separada para instruções e dados (operandos) e, portanto, duas endereços e dois barramentos de dados

Æ busca paralela de instrução(ões) e operando(s) possível

- Versões simplificadas

1. Dois sistemas de barramento separados até os caches L1, mas apenas um principal memória/cache L2/L3 unificado (mais barato, padrão nos sistemas atuais)
2. Apenas um único sistema de barramento multiplexado

54

CISC / RISC

- Unidade de controle
 - Com fio
 - O registro de instrução é uma fila FIFO simples
 - Cada estágio do pipeline tem seu próprio registrador • Um circuito combinacional simples pode "interpretar" os OpCodes em cada estágio diretamente
- Registrar arquivo
 - Consiste em um grande número de registros (de uso geral)
 - Suporta a seleção simultânea de vários registradores
 - Por exemplo, arquivo de registro de 4 portas: gravação simultânea em R0, R1 e leitura de R2, R3

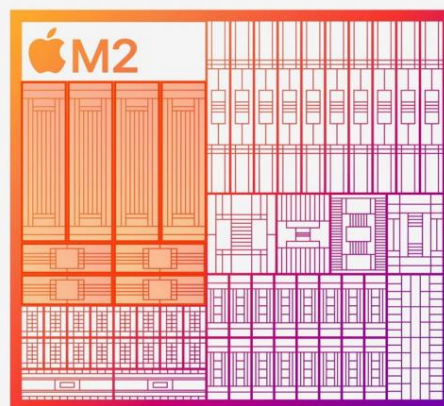
55

4 high-performance cores

Ultrawide microarchitecture
192KB instruction cache
128KB data cache
Shared 16MB cache

4 high-efficiency cores

Wide microarchitecture
128KB instruction cache
64KB data cache
Shared 4MB cache



56