Arquitetura de Computadores
Aula 11

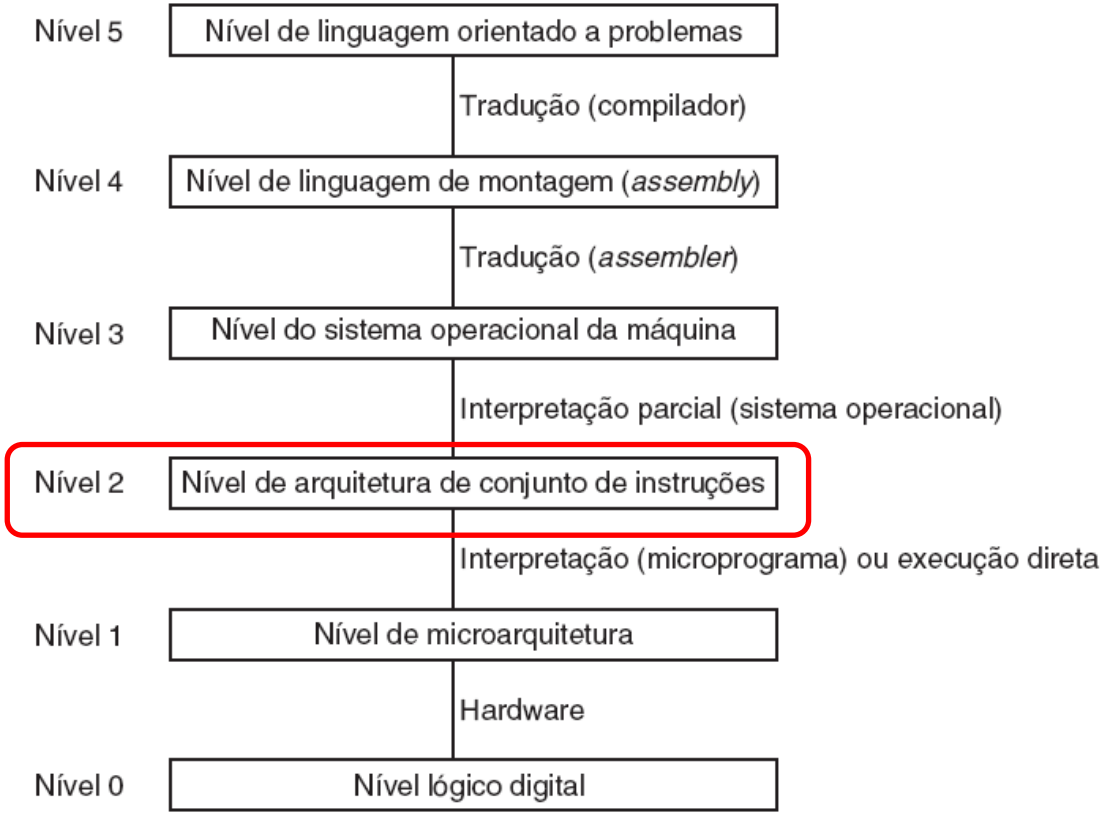
Capítulo 5

Capítulo 5

Nível da arquitetura do conjunto de instruções
ISA

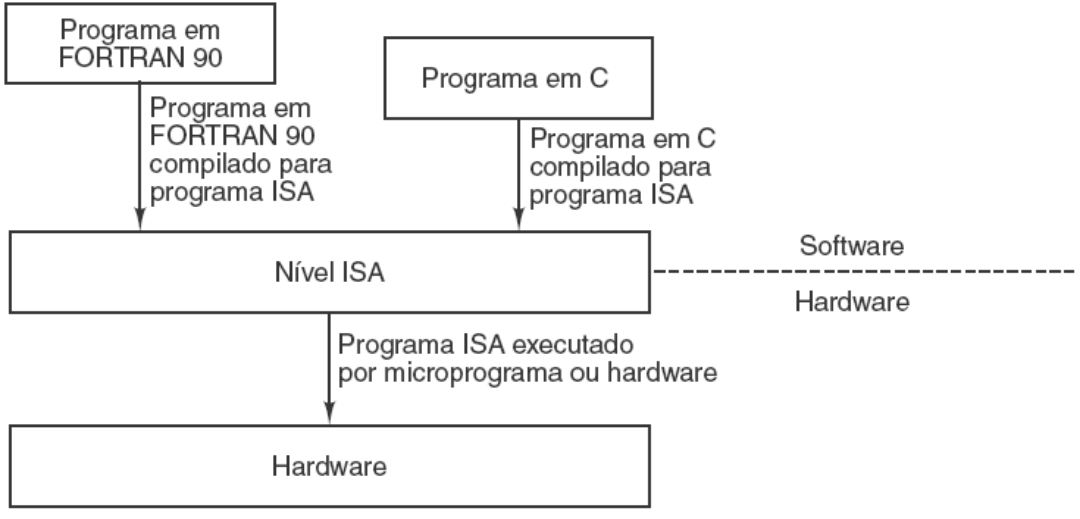
5 Nível de arquitetura do conjunto de instrução

Nível ISA



Nível ISA

- Interface entre o software e o hardware.
- Define a interface entre os compiladores e o hardware. É a linguagem que ambos devem entender.



O que faz uma ISA ser boa?

- Em primeiro lugar: uma boa ISA deve definir um conjunto de instruções que pode ser implementado com eficiência em tecnologias atuais e futuras, resultando em projetos de efetivos em custo por várias gerações.

- Em segundo lugar: deve fornecer um alvo claro para o código compilado.

Regularidade e completude de uma faixa de opções são aspectos importantes que nem sempre estão presentes em umas ISA.

5 Nível de arquitetura do conjunto de instrução

ISA: interface entre o hardware o software

- Deve contentar os projetistas de hardware: fácil de implementar com eficiência
- Deve contentar os projetistas de software: fácil de gerar bom código para ela

5.2 Tipos de dados

- Uma questão fundamental: há ou não suporte de hardware para um tipo particular de dados.

- Suporte de hardware: significa que uma ou mais instruções espera dados em um formato particular e o usuário não tem liberdade de escolher um formato diferente.

5.2.1 Tipos de dados numéricos

- Divisão dos tipos de dados: Numéricos e Não Numéricos

Numérico

- Inteiro
  - comprimento 8,16, 32 e 64 bits
  - contam coisas e identificam coisas
  - computadores podem suportar inteiros com sinal ou sem sinal

- Ponto flutuante
  - comprimento 32, 64 ou 128 bits

✓ Muitos computadores tem registradores separados para conter operandos inteiros e operandos de ponto flutuante

5.2.2 Tipos de dados não numéricos

- Ex. de aplicações não numéricas: e-mail, fotografia digital, reprodução multimídia

- Caracteres
  - ASCII (7 bits)
  - UNICODE (16 bits)
  - Cadeias de caracteres (delimitadas por um caractere especial na extremidade)
  - Valores booleanos (pode assumir 2 valores)
    - Mapas de bits
  - Ponteiro: endereço de máquina (ex. SP e LV)

5.2.3 Tipos de dados no Pentium 4

Tipo	1 bit	8 bits	16 bits	32 bits	64 bits	128 bits
Bit						
Inteiro com sinal		×	×	×		
Inteiro sem sinal		×	×	×		
Inteiro decimal em código binário		×				
Ponto flutuante				×	×	

- Há instruções especiais para copiar e buscar cadeias de caracteres

5.2.4 Tipos de dados na UltraSPARC III

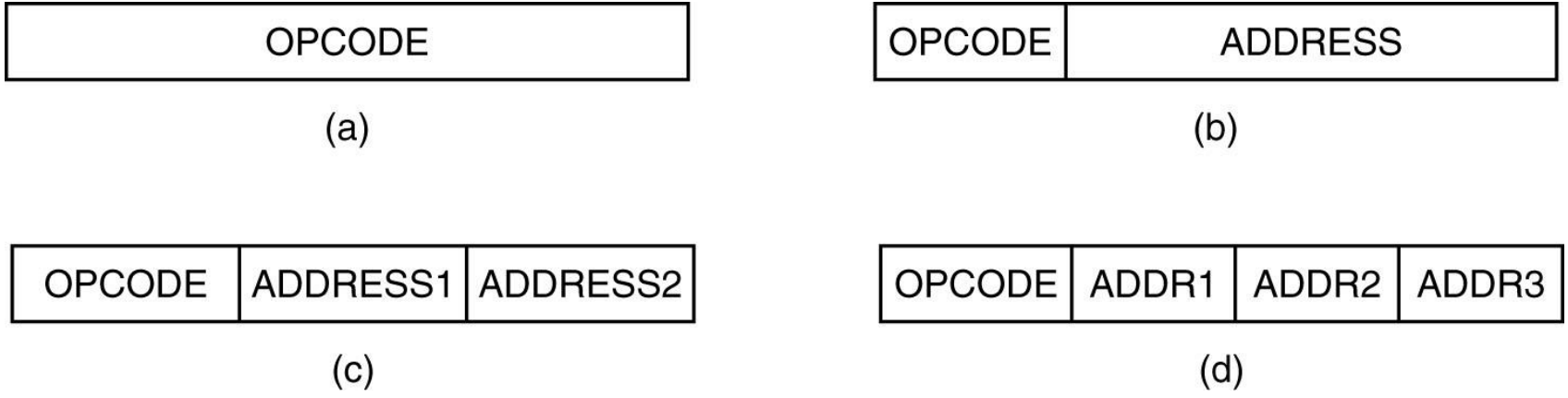
Tipo	1 bit	8 bits	16 bits	32 bits	64 bits	128 bits
Bit						
Inteiro com sinal		×	×	×	×	
Inteiro sem sinal		×	×	×	×	
Inteiro decimal em código binário						
Ponto flutuante				×	×	×

- Tipos de dados de caracteres e de cadeia não são suportados por instruções especiais de hardware: são manipulados inteiramente em software

5.2.5 Tipos de dados do 8051

Tipo	1 bit	8 bits	16 bits	32 bits	64 bits	128 bits
Bit	×					
Inteiro com sinal		×				
Inteiro sem sinal						
Inteiro decimal em código binário						
Ponto flutuante						

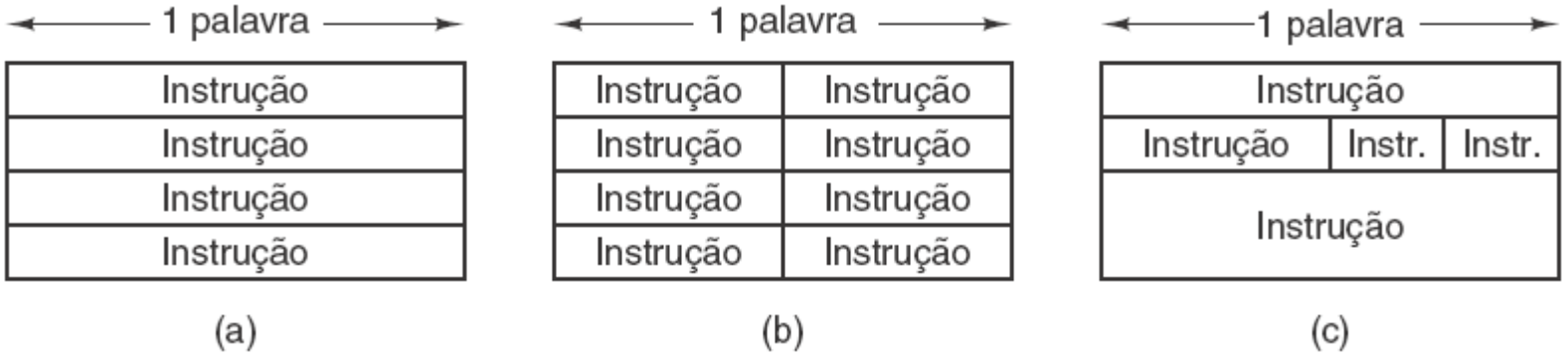
5.3 Formatos de instrução



Quatro formatos comuns de instrução:

(a) Instrução sem endereço.
(b) Instrução com um endereço.
(c) Instrução com dois endereços.
(d) Instrução com três endereços.

5.3 Formatos de instrução



Algumas relações possíveis entre comprimento de instrução e de palavra.

5.3.1 Critérios de projeto para formatos de instruções

- Tamanho das instruções
  - Instruções curtas são melhores que longas (porém minimizar instruções pode torna-las mais difíceis de codificar)
- Espaço suficiente no formato da instrução para expressar todas as operações desejadas
- Número de bits em um campo de endereço

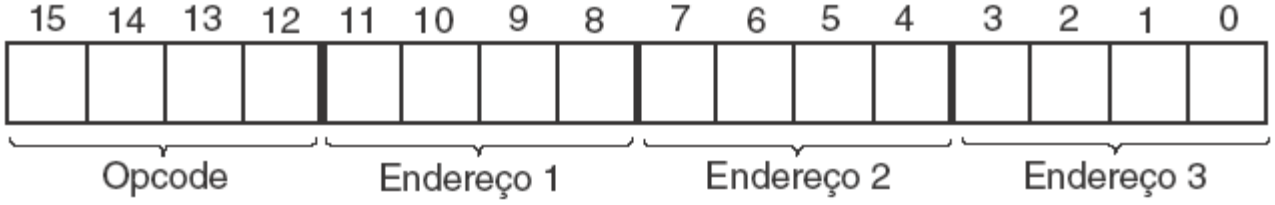
5.3.2 Expansão de opcodes

- Instrução de (n + k) bits -> opcode de k bits e um único endereço de n bits
  - 2^n operações diferentes e 2^n células de memória endereçáveis.

  - (k – 1) bits -> opcode e (n + 1) endereços:
    - ✓ Metade do número de instruções, mas duas vezes mais memória endereçável

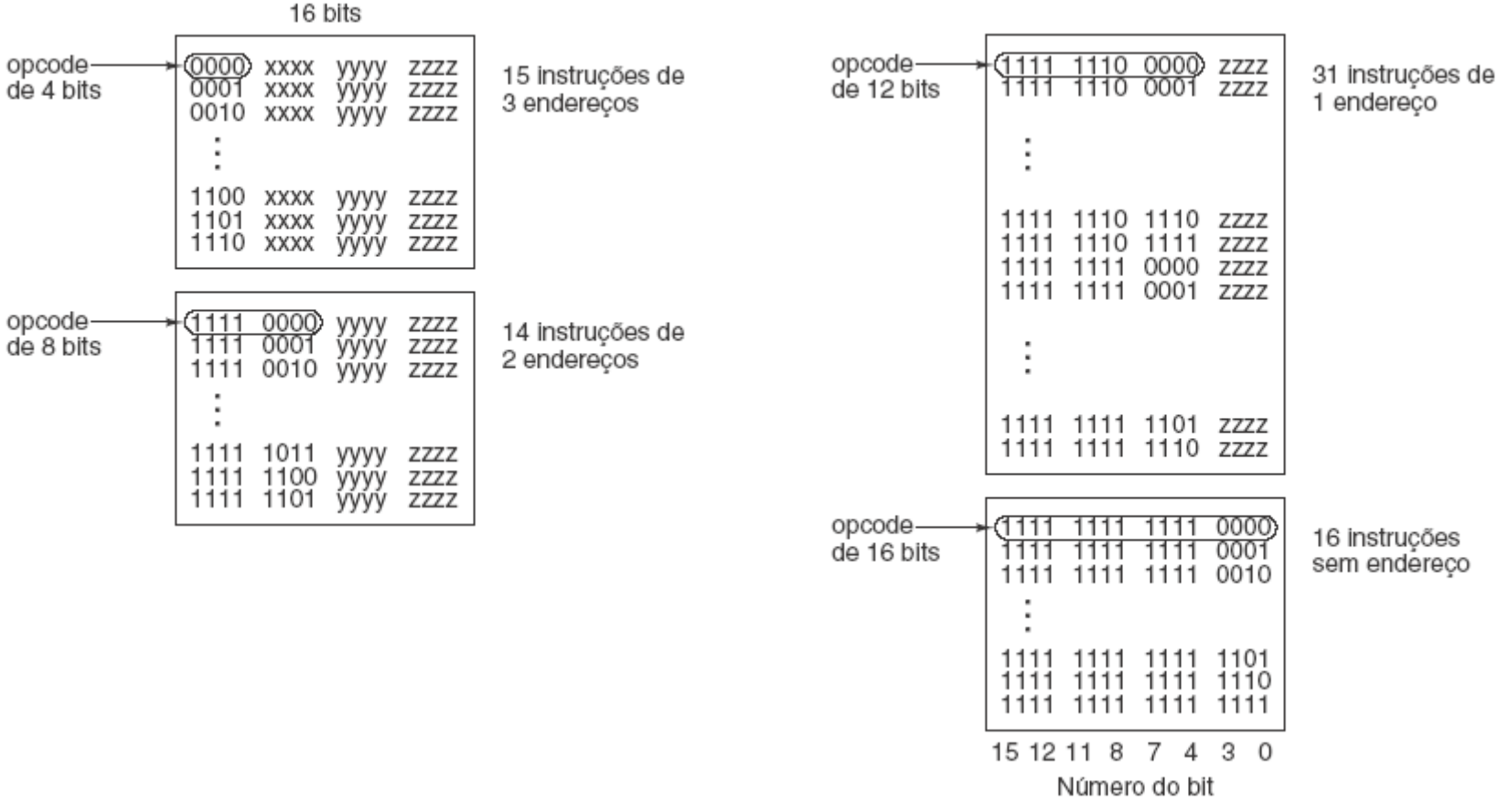
  - (k + 1) bits -> opcode e (n - 1) endereços:
    - ✓ Mais operações, menor número de células endereçáveis

5.3.2 Expansão de opcodes

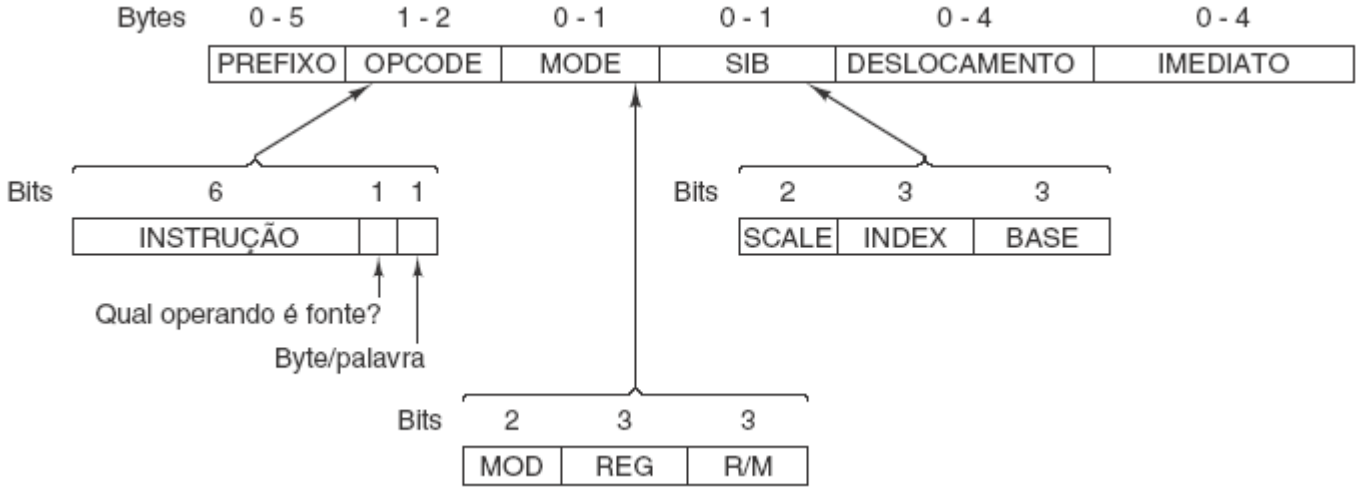


Instrução com um opcode de 4 bits e três campos de endereço de 4 bits cada.

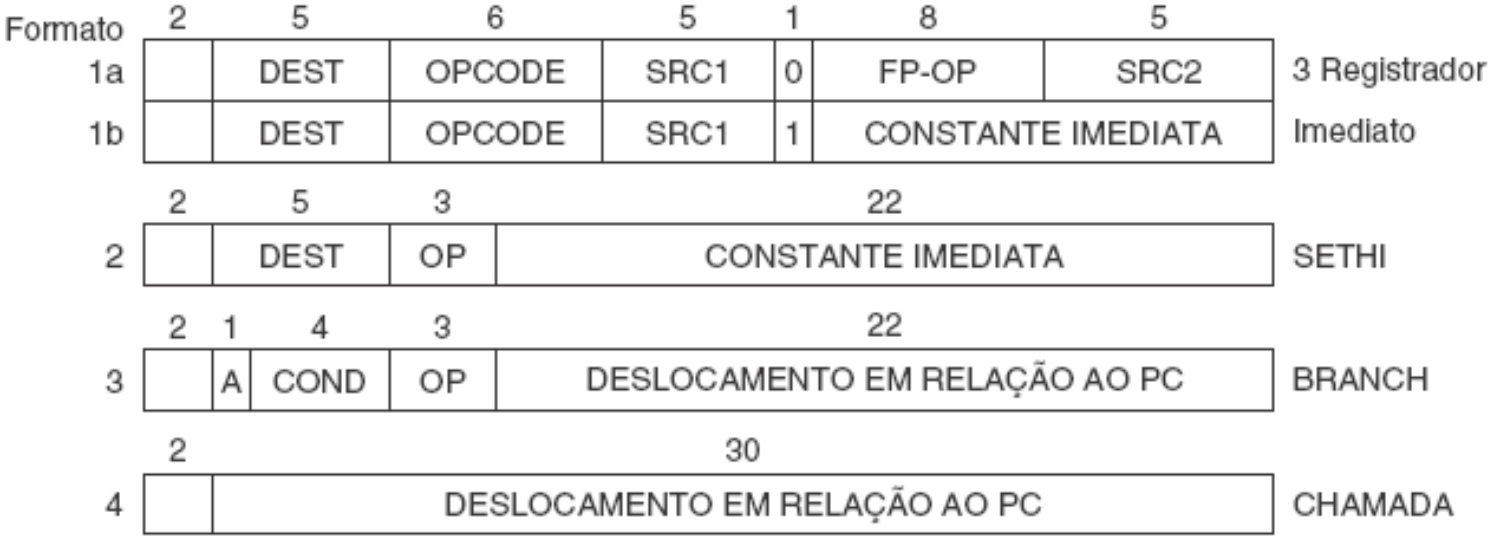
5.3.2 Expansão de opcodes



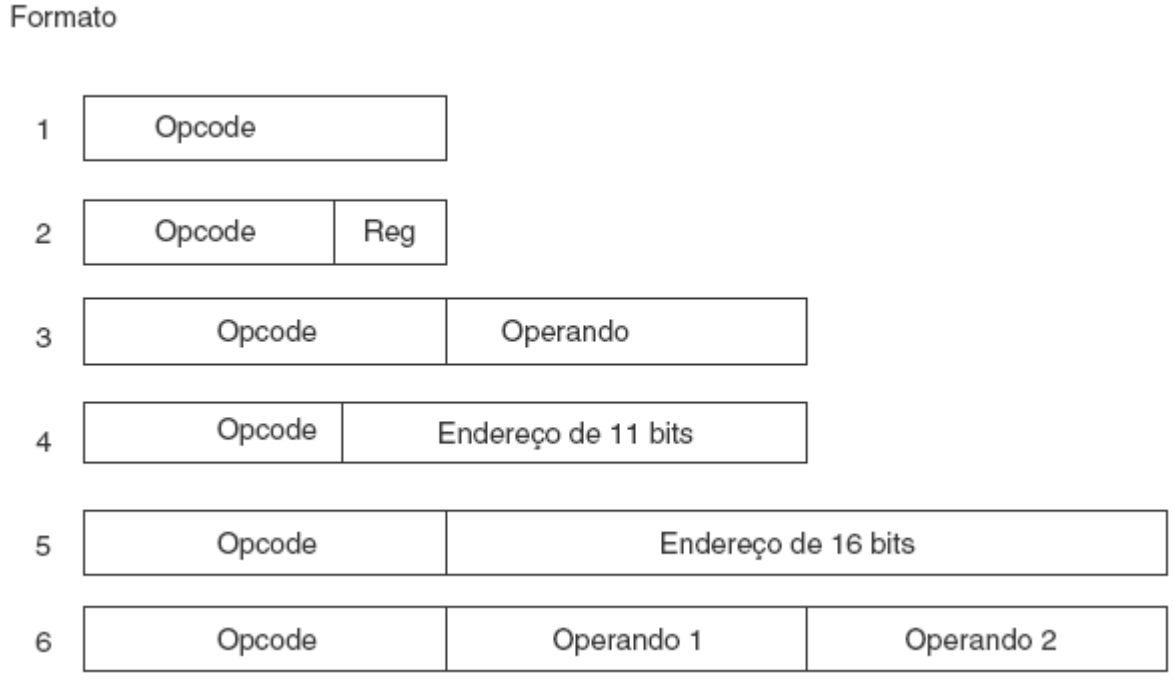
5.3.3 Formato de instruções do Pentium 4



5.3.4 Formato de instruções da UltraSPARC III



5.3.4 Formato de instruções do 8051



5.4 Endereçamento

- Modos de endereçamento:
  - Endereçamento imediato
  - Endereçamento direto
  - Endereçamento de registrador
  - Endereçamento indireto de registrador
  - Endereçamento indexado
  - Endereçamento de base indexado
  - Endereçamento de pilha

5.4.2 Endereçamento Imediato

- É a parte da instrução referente ao endereço realmente contem o operando em si

Instrução imediata para carregar 4 no registrador 1.

Vantagem: não exige uma referência extra à memória para buscar o operando
Desvantagem: somente uma constante pode ser fornecida desse modo e o número de valores é limitado pelo tamanho do campo

5.4.2 Endereçamento Direto

- Método para especificar um operando na memória dando seu endereço completo
- Instrução sempre acessará exatamente a mesma localização de memória
- Só pode ser usado para acessar variáveis globais cujos endereços são conhecidos na hora da compilação

5.4.2 Endereçamento de registrador

- Conceitualmente o mesmo que endereçamento direto
- Especifica uma registrador em vez de uma localização de memória

5.4.5 Endereçamento indireto de registrador

- Endereço do dado é obtido do conteúdo da posição identificada pela instrução (endereço na instrução = ponteiro)
  - ° Referencia memória sem precisar de endereço de memória completo na instrução

- Ex: soma elementos de um vetor de 1024 inteiros de 4 bytes cada

```
MOV R1, #0                  ; Acumula soma em R1, inicialmente 0
MOV R2, #A                  ; R2 = endereço de vetor A
MOV R3, #A + 4096           ; R3 = endereço da 1a palavra depois de A
LOOP: ADD R1, (R2)          ; Indireto de registrador via R2 para obter operando
ADD R2, #4                  ; Incrementa R2 de uma palavra (imediato)
CMP R2, R3                  ; Já terminou?
BLT LOOP                    ; R2 < R3 : continue
```

5.4.6 Endereçamento indexado

- Endereçamento indexado é o nome que se dá ao endereçamento de memória que fornece um registrador (explícito ou implícito) mais um deslocamento constante

Programa genérico em linguagem de montagem para calcular a operação OR de A_i AND B_i para dois vetores de 1024 elementos.

```
        MOV R1,#0        ; Acumula o OR em R1, inicialmente 0
        MOV R2,#0        ; R2 = índice, i, do produto atual: A[i] AND B[i]
        MOV R3,#4096     ; R3 = primeiro valor do índice que não deve ser usado
LOOP:   MOV R4,A(R2)     ; R4 = A[i] ]
        AND R4,B(R2)     ; R4 = A[i] AND B[i]
        OR R1,R4         ; Faz o OR do produto booleano com o conteúdo de R1
        ADD R2,#4        ; i = 1 + 4 (pule em unidade de palavra = 4 bytes)
        CMP R2,R3        ; Já terminamos?
        BLT LOOP         ; Se R2 < R3, não terminamos, portanto continue
```

5.4.6 Endereçamento indexado

- O endereço do dado é obtido pela combinação de dois valores
  - ° Endereço de base: constante (ou registrador)
  - ° Offset: deslocamento a partir da base

- Útil no acesso a vetores

- MOV R1, A(R3)
  - ° O dado a ser colocado em R1 está em uma posição de memória obtida ao se somar o endereço inicial do vetor A (uma constante) com o valor armazenado em R3 (pode ser incrementado/decrementado)

5.4.7 Endereçamento de base indexado

- Endereço do dado é obtido pela soma dos valores de dois registradores e opcionalmente uma constante

- MOV R1, (R3+R4)
  - ➢Os valores de R3 e R4 são somados e o valor resultante é usado para acessar uma posição de memória de onde o novo valor de R1 é lido

- Para o exemplo anterior teríamos:

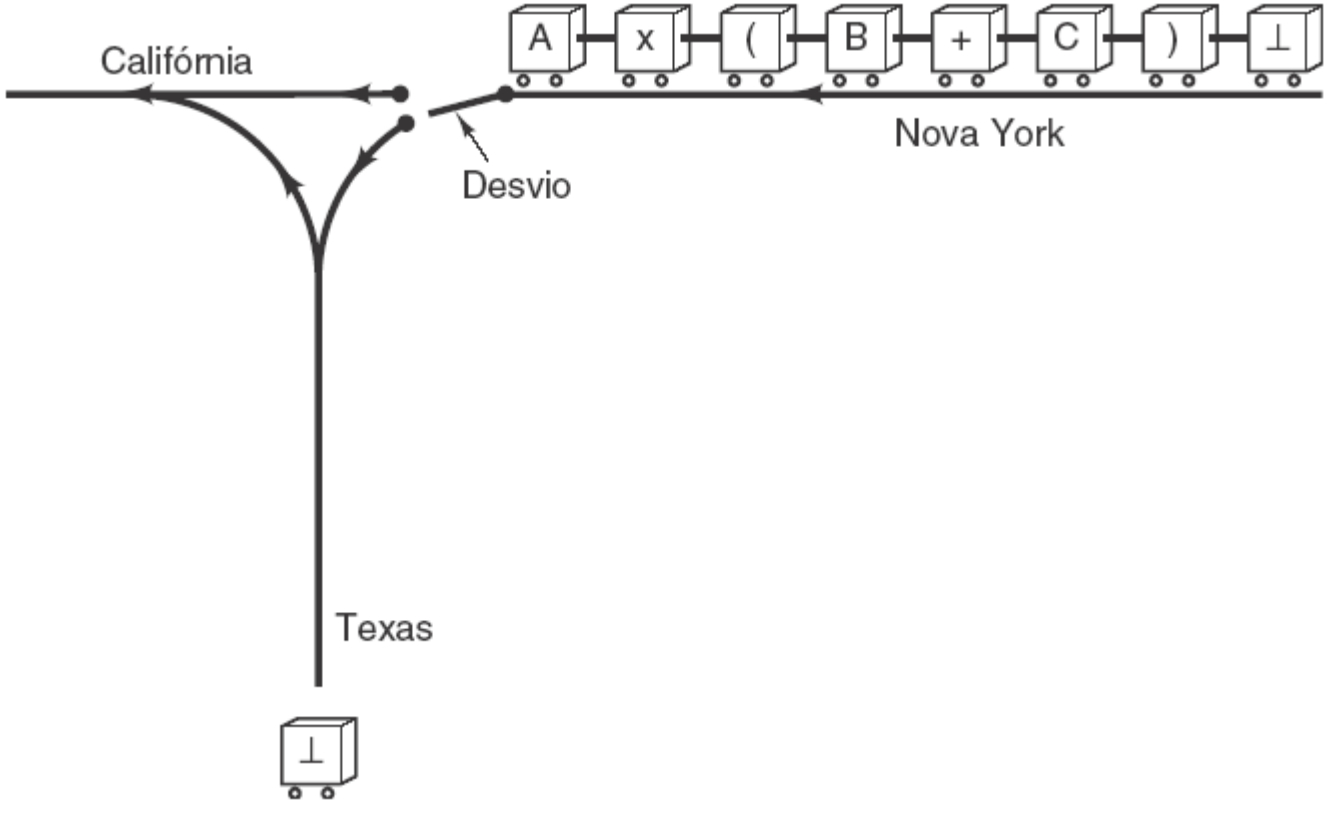
```
LOOP:  MOV R4,(R2+R5)
       MOV R4,(R2+R6)
```

5.4.8 Endereçamento de pilha

- Instruções de máquinas devem ser as mais curtas possíveis
- Limite: nenhum endereço é fornecido na instrução (endereçamento implícito)
- O processador tem a noção de manipulação de dados em uma pilha
- Operações sempre se referem ao topo da pilha em qualquer instante

5.4.8 Endereçamento de pilha

Notação Polonesa Invertida



5.4.8 Endereçamento de pilha

Notação Polonesa Invertida

1. O vagão que está no desvio vai para Texas
2. O vagão mais recente na linha do Texas faz o retorno e vai para a Califórnia
3. O vagão que está no desvio e o vagão mais recente na linha do Texas são desviados e desaparecem (apagados)
4. Pare. Os símbolos agora em Califórnia representam a fórmula em notação polonesa invertida quando lida da esquerda para a direita
5. Pare. Ocorreu um erro. A fórmula original não foi equilibrada adequadamente

Vagão que chegou mais recentemente à linha do Texas \ Vagão no desvio	⊥	+	−	x	/	(	)
⊥	4	1	1	1	1	1	5
+	2	2	2	1	1	1	2
−	2	2	2	1	1	1	2
x	2	2	2	2	2	1	2
/	2	2	2	2	2	1	2
(	5	1	1	1	1	1	3

Tabela de decisão usada pelo algoritmo de conversão da notação infixa para a notação polonesa invertida

5.4.8 Endereçamento de pilha

Alguns exemplos de expressões infixas e seus equivalentes em notação polonesa invertida:

Infix	Reverse Polish notation
A + B × C	A B C × +
A × B + C	A B × C +
A × B + C × D	A B × C D × +
(A + B) / (C – D)	A B + C D – /
A × B / C	A B × C /
((A + B) × C + D)/(E + F + G)	A B + C × D + E F + G + /

5.4.8 Endereçamento de pilha

Utilização de uma pilha para avaliar uma fórmula em notação polonesa invertida:

Etapa	Cadeia restante	Instrução	Pilha
1	8 2 5 × + 1 3 2 ×+ 4 − /	BIPUSH8	8
2	2 5 × + 1 3 2 × + 4 − /	BIPUSH2	8, 2
3	5 × + 1 3 2 × + 4 − /	BIPUSH5	8, 2, 5
4	× + 1 3 2 × + 4 − /	IMUL	8, 10
5	+ 1 3 2 × + 4 − /	IADD	18
6	1 3 2 × + 4 − /	BIPUSH1	18, 1
7	3 2 × + 4 − /	BIPUSH3	18, 1, 3
8	2 × + 4 − /	BIPUSH2	18, 1, 3, 2
9	× + 4 − /	IMUL	8, 1, 6
10	+ 4 − /	IADD	18, 7
11	4 − /	BIPUSH4	18, 7, 4
12	− /	ISUB	18, 3
13	/	IDIV	6