Codificando as instruções em binário

Prof. Eduardo H. M. Cruz

Instruções

- A linguagem Assembly é apenas uma abstração humana da linguagem de máquina (código binário)

- Logo, há uma forma de traduzir de assembly para binário
  - E o contrário também, de binário para assembly

- Para simplificar o aprendizado, consideraremos neste momento um formato de instruções do tipo RISC
  - RISC: Reduced instruction set computer
  - Todas as instruções tem o mesmo número de bits
  - Formato simples
  - Campos em posições bem definidas

Instruções

- Usaremos aqui um formato hipotético de instruções de 16 bits
- Nosso conjunto de instruções terá 2 tipos de formato
  - Formato R: instruções com operandos em registradores
    - Em sua maioria, instruções da ULA
  - Formato I: instruções com operandos imediatos
    - Mover dados imediatos e saltos
- O bit 15 determina o formato
  - Se for 0: formato R
  - Se for 1: formato I
- Usaremos 3 bits para identificar cada registrador

Instruções

Formato R

Formato	Opcode	Destino	Operando 1	Operando 2
1 bit	6 bits	3 bits	3 bits	3 bits

Campo	Bits (posição)	Explicação
Formato	15	Vai ter valor 0, para identificar que é do tipo R
Opcode	9-14	Código da operação
Destino	6-8	Registrador de destino
Operando 1	3-5	Registrador com o operando 1
Operando 2	0-2	Registrador com o operando 2

Instruções

Formato R

Formato	Opcode	Destino	Operando 1	Operando 2
1 bit	6 bits	3 bits	3 bits	3 bits

- Exemplo: add r5, r6, r1 (opcode do add é 0)
  - Formato: 0 (tipo R)
  - Opcode: 000000 (0 em decimal, add)
  - Destino: 101 (5 em decimal)
  - Operando 1: 110 (6 em decimal)
  - Operando 2: 001 (1 em decimal)

- Instrução codificada: 0 000000 101 110 001

Exercício

- Supondo que a instrução de subtração use o formato R o tenha o opcode 1 em decimal, codifique a seguinte instrução de linguagem de montagem para sua representação binária

- Sub r7, r7, r2

Instruções

Formato R

Formato	Opcode	Destino	Operando 1	Operando 2
1 bit	6 bits	3 bits	3 bits	3 bits

- sub r7, r7, r2
  - Formato: 0 (tipo R)
  - Opcode: 000001 (1 em decimal, sub)
  - Destino: 111 (7 em decimal)
  - Operando 1: 111 (7 em decimal)
  - Operando 2: 010 (2 em decimal)

- Instrução codificada: 0 000001 111 111 010

Codificando load

Formato R, operando 1 conterá endereço

Formato	Opcode	Destino	Operando 1	Operando 2
1 bit	6 bits	3 bits	3 bits	3 bits

- load r7, [r2]
  - Formato: 0 (tipo R)
  - Opcode: 001111 (15 em decimal, load)
  - Destino: 111 (7 em decimal)
  - Operando 1: 010 (2 em decimal)
  - Operando 2: **ignoramos** pois não é usado, usaremos 000 então

  - **Endereço de memória fica no Op1**

- Instrução codificada: 0 001111 111 010 000

Codificando store

Formato R, operando 1 conterá o endereço

Formato	Opcode	Destino	Operando 1	Operando 2
1 bit	6 bits	3 bits	3 bits	3 bits

- store [r7], r2
  - Formato: 0 (tipo R)
  - Opcode: 010000 (16 em decimal, load)
  - Destino: **ignoramos** pois não é usado, usaremos 000 então
  - Operando 1: 111 (7 em decimal)
  - Operando 2: 010 (2 em decimal)

  - **Endereço de memória fica no Op1**
  - **Valor a ser escrito fica no Op2**

- Instrução codificada: 0 010000 000 111 010

Instruções

DICA:
Usem um conversor decimal-binário para gerar o imediato

Formato I

Formato	Opcode	Registrador	Imediato
1 bit	2 bits	3 bits	10 bits

Campo	Bits (posição)	Explicação
Formato	15	Vai ter valor 1, para identificar que é do tipo I
Opcode	13-14	Código da operação
Registrador	10-12	Registrador contendo um operando
Imediato	0-9	Valor imediato contido na instrução

Instruções

Formato I

Formato	Opcode	Registrador	Imediato
1 bit	2 bits	3 bits	10 bits

- Exemplo: mov r3, 57 (opcode do mov é 3)
  - Formato: 1 (tipo I)
  - Opcode: 11 (3 em decimal, mov)
  - Registrador: 011 (3 em decimal)
  - Imediato: 00 0011 1001 (57 em decimal)

- Instrução codificada: 1 11 011 00 0011 1001

Exercício

- Codifique em binário a seguinte instrução
- Mov r0, 129

Instruções

Formato I

Formato	Opcode	Registrador	Imediato
1 bit	2 bits	3 bits	10 bits

- Exemplo: mov r0, 129 (opcode do mov é 3)
  - Formato: 1 (tipo I)
  - Opcode: 11 (3 em decimal, mov)
  - Registrador: 000 (0 em decimal)
  - Imediato: 00 1000 0001 (129 em decimal)

- Instrução codificada: 1 11 000 00 1000 0001

Codificando instruções de salto em binário

- Usaremos o formato I

Formato	Opcode	Registrador	Imediato
1 bit	2 bits	3 bits	10 bits

- Opcode:
  - 0: jump
  - 1: jump_cond
- Registrador
  - Jump: conteúdo é ignorado
  - Jump_cond: contém o registrador que indicará se irá saltar ou não
- Imediato
  - Endereço alvo do salto

Conjunto de instruções da nossa arquitetura

Instruções do formato R (com operandos em registradores)

Formato	Opcode	Mnemônico	Explicação
0	000000 (0)	Add	Adição de registradores
0	000001 (1)	Sub	Subtração de registradores
0	000010 (2)	mul	Multiplicação de registradores
0	000011 (3)	div	Divisão de registradores
0	000100 (4)	cmp_equal	Compara se operandos op1 = op2
0	000101 (5)	Cmp_neq	Compara se operandos op1 != op2
0	000110 (6)	Cmp_less	Compara se op1 < op2
0	000111 (7)	Cmp_greater	Compara se op1 > op2
0	001000 (8)	Cmp_less_eq	Compara se op1 <= op2
0	001001 (9)	Cmp_greater_eq	Compara se op1 >= op2

Conjunto de instruções da nossa arquitetura

Instruções do formato R (com operandos em registradores)

Formato	Opcode	Mnemônico	Explicação
0	001010 (10)	and	Operação AND entre registradores
0	001011 (11)	or	Operação OR entre registradores
0	001100 (12)	Xor	Operação XOR entre registradores
0	001101 (13)	Shl	Deslocar bits para a esquerda
0	001110 (14)	Shr	Deslocar bits para a direita
0	001111 (15)	Load	Carregar da memória para o registrador
0	010000 (16)	Store	Salvar na memória o conteúdo de um registrador
0	...	...	...
0	...	...	...
0	111111 (63)	...	...

Conjunto de instruções da nossa arquitetura

Instruções do formato I (com operandos imediatos)

Formato	Opcode	Mnemônico	Explicação
1	00 (0)	Jump	Salto incondicional
1	01 (1)	Jump_cond	Salto condicional
1	10 (2)	---	Não usado
1	11 (3)	Mov	Move um imediato para um registrador

Exercício

- Para o seguinte trecho de código assembly, gere sua codificação binária

- Mov r5, 1
- And r0, r0, r5
- Jump_cond r0, 48
- Add r5, r5, r5
- Load r2, [r4]
- Jump 50

Exercício

- mov r5, 1
  - Formato: 1 (tipo I)
  - Opcode: 11 (3 em decimal, mov)
  - Registrador: 101 (5 em decimal)
  - Imediato: 00 0000 0001 (1 em decimal)

- Instrução codificada: 1 11 101 00 0000 0001

Exercício

- And r0, r0, r5
  - Formato: 0 (tipo R)
  - Opcode: 001010 (10 em decimal, and)
  - Destino: 000 (0 em decimal)
  - Operando 1: 000 (0 em decimal)
  - Operando 2: 101 (5 em decimal)

- Instrução codificada: 0 001010 000 000 101

Exercício

- Jump_cond r0, 48
  - Formato: 1 (tipo I)
  - Opcode: 01 (1 em decimal, jump_cond)
  - Registrador: 000 (0 em decimal)
  - Imediato: 00 0011 0000 (48 em decimal)

- Instrução codificada: 1 01 000 00 0011 0000

Exercício

- Add r5, r5, r5
  - Formato: 0 (tipo R)
  - Opcode: 000000 (0 em decimal, add)
  - Destino: 101 (5 em decimal)
  - Operando 1: 101 (5 em decimal)
  - Operando 2: 101 (5 em decimal)

- Instrução codificada: 0 000000 101 101 101

Exercício

- Load r2, [r4]
  - Formato: 0 (tipo R)
  - Opcode: 001111 (15 em decimal, load)
  - Destino: 010 (2 em decimal)
  - Operando 1: 100 (4 em decimal)
  - Operando 2: 000 (campo não usado)

- Instrução codificada: 0 001111 010 100 000

Exercício

- Jump 50
  - Formato: 1 (tipo I)
  - Opcode: 00 (0 em decimal, jump)
  - Registrador: pode por qualquer valor, usaremos 000
  - Imediato: 00 0011 0010 (50 em decimal)

- Instrução codificada: 1 00 000 00 0011 0010

Exercício

Código assembly	Código binário
Mov r5, 1	1 11 101 00 0000 0001
And r0, r0, r5	0 001010 000 000 101
Jump_cond r0, 48	1 01 000 00 0011 0000
Add r5, r5, r5	0 000000 101 101 101
Load r2, [r4]	0 001111 010 100 000
Jump 50	1 00 000 00 0011 0010