Arquiteturas

Introdução a Arquiteturas de Computadores 2/3

Glauber De Bona

PCS - Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

Setembro, 2020

Agenda

Codificação
Lógicas
Imediatos
Desvio
Exercícios

1 Codificação

2 Lógicas

3 Imediatos

4 Desvio

5 Exercícios

Codificando Instruções

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Considere a seguinte instrução:

`ADD X9,X20,X21`

Como esta instrução é representada na memória?

1112	21	0	20	9

10001011000	10101	000000	10100	01001
11 bits	5 bits	6 bits	5 bits	5 bits

Codificação de Instruções no LEGv8

Codificação
Lógicas
Imediatos
Desvio
Exercícios

- A instrução em binário é exatamente o que é armazenado na memória.
- Quando o processador ler a instrução, irá decodificá-la e ligar os sinais necessários para que executá-la.
- As instruções do LEGv8 tem sempre 32 bits.
- Há seis formatos de instruções diferentes.

Formato R

Codificação
Lógicas
Imediatos
Desvio
Exercícios

- Formato usado em instruções com operandos e destino em registradores.
- Campos (e.g. ADD X1,X2,X3)
  - *opcode:* o código que representa a instrução (ADD=1112).
  - *Rm:* O segundo registrador para a operação (e.g. 3).
  - *shamt: Shift Ammount*, a quantidade de *shift* a ser feita antes de gravar o resultado.
  - *Rn:* O primeiro registrador para a operação (e.g. 2).
  - *Rd:* O destino do resultado da operação (e.g. 1).

Formato D

Codificação
Lógicas
Imediatos
Desvio
Exercícios

opcode	address	op2	Rn	Rt
11 bits	9 bits	2 bits	5 bits	5 bits

- Formato usado em instruções de transferência de dados (*load/store*).
- Campos (e.g. LDUR X1,[X2,#16]):
  - *opcode,op2:* o código que representa a instrução (e.g. LDUR=1986,0).
  - *address:* O número que será somado ao registrador base para calcular o endereço (e.g. 16).
  - *Rn:* O registrador base (e.g. 2).
  - *Rt:* O destino ou fonte (*target*) da operação (e.g. 1).

Formato I

Codificação
Lógicas
Imediatos
Desvio
Exercícios

- Formato usado em instruções onde um dos operandos é um imediato.
- Campos (e.g. ADDI X1,X2,#58):
  - *opcode:* o código que representa a instrução (e.g. ADDI=580).
  - *immediate:* O número que será usado no lugar de um dos operandos (e.g. 58).
  - *Rn:* O primeiro registrador para a operação (e.g. 2).
  - *Rd:* O destino do resultado da operação (e.g. 1).

Resumo de *opcodes*

Instruction	Format	opcode	Rm	shamt	address	op2	Rn	Rd
ADD (add)	R	1112_{ten}	reg	0	n.a.	n.a.	reg	reg
SUB (subtract)	R	1624_{ten}	reg	0	n.a.	n.a.	reg	reg
ADDI (add immediate)	I	580_{ten}	reg	n.a.	constant	n.a.	reg	n.a.
SUBI (sub immediate)	I	836_{ten}	reg	n.a.	constant	n.a.	reg	n.a.
LDUR (load word)	D	1986_{ten}	reg	n.a.	address	0	reg	n.a.
STUR (store word)	D	1984_{ten}	reg	n.a.	address	0	reg	n.a.

Exemplo

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Considere um vetor A de doublewords:

```
1  A[30] = h + A[30] + 1;
```

O código compilado para o LEGv8 será:

```
1  LDUR X9, [X10,#240]
2  ADD X9,X21,X9
3  ADDI X9,X9,#1
4  STUR X9, [X10,#240]
```

Exemplo

Codificação
Lógicas
Imediatos
Desvio
Exercícios

```
1  LDUR X9, [X10,#240]
2  ADD X9,X21,X9
3  ADDI X9,X9,#1
4  STUR X9, [X10,#240]
```

opcode	Rm/address	shamt/op2	Rn	Rd/Rt
1986	240	0	10	9
1112	9	0	21	9
580	1		9	9
1984	240	0	10	9

Exemplo


```
1  LDUR X9, [X10,#240]
2  ADD X9,X21,X9
3  ADDI X9,X9,#1
4  STUR X9, [X10,#240]
```

1111100001 0	011110000	00	01010	01001
10001011000	01001	000000	10101	01001
1001000100	000000000001		01001	01001
1111100000 0	011110000	00	01010	01001

Resumo da Codificação

Name	Format	Example						Comments
ADD	R	1112	3	0		2	1	ADD X1, X2, X3
SUB	R	1624	3	0		2	1	SUB X1, X2, X3
ADDI	I	580	100			2	1	ADDI X1, X2, #100
SUBI	I	836	100			2	1	SUBI X1, X2, #100
LDUR	D	1986	100		0	2	1	LDUR X1, [X2, #100]
STUR	D	1984	100		0	2	1	STUR X1, [X2, #100]
Field size		11 or 10 bits	5 bits	5 or 4 bits	2 bits	5 bits	5 bits	All ARM instructions are 32 bits long
R-format	R	opcode	Rm	shamt		Rn	Rd	Arithmetic instruction format
I-format	I	opcode	immediate			Rn	Rd	Immediate format
D-format	D	opcode	address		op2	Rn	Rt	Data transfer format

Exemplo 2

Name	Format	Example						Comments
ADD	R	1112	3	0		2	1	ADD X1, X2, X3
SUB	R	1624	3	0		2	1	SUB X1, X2, X3
ADDI	I	580	100			2	1	ADDI X1, X2, #100
SUBI	I	836	100			2	1	SUBI X1, X2, #100
LDUR	D	1986	100		0	2	1	LDUR X1, [X2, #100]
STUR	D	1984	100		0	2	1	STUR X1, [X2, #100]
Field size		11 or 10 bits	5 bits	5 or 4 bits	2 bits	5 bits	5 bits	All ARM instructions are 32 bits long
R-format	R	opcode	Rm	shamt		Rn	Rd	Arithmetic instruction format
I-format	I	opcode	immediate			Rn	Rd	Immediate format
D-format	D	opcode	address		op2	Rn	Rt	Data transfer format

Qual instrução está representada abaixo?

opcode	Rm	shamt	Rn	Rd
1624	9	0	10	11

Instruções Lógicas

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Para manipular bit individuais dentro dos registradores, temos instruções lógicas:

Logical operations	C operators	Java operators	LEGv8 instructions
Shift left	<<	<<	LSL
Shift right	>>	>>>	LSR
Bit-by-bit AND	&	&	AND,ANDI
Bit-by-bit OR	|	|	OR,ORI
Bit-by-bit NOT	~	~	EOR,EORI

Por exemplo, ANDs podem zerar bits selecionados (máscara).

Instruções Lógicas (Exemplos)

Codificação
Lógicas
Imediatos
Desvio
Exercícios

```
1  LSR X11,X19,#4 // reg X11 = reg X19 >> 4 bits
2  LSL X11,X19,#4 // reg X11 = reg X19 << 4 bits
3  AND X9,X10,X11 // reg X9 = reg X10 & reg X11
4  ORR X9,X10,X11 // reg X9 = reg X10 | reg X11
5  EOR X9,X10,X12 // reg X9 = reg X10 xor reg X12
```

Instruções Lógicas (Exercício)

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Qual o resultado de X9 após cada uma destas instrucões?

```
1   EORI  X9, X9, #0xFFF
2   EOR   X9, X9, X9
```

Na primeira, X9=not(X9), na segunda, X9=zero.

Instruções Lógicas - Codificação

Codificação
Lógicas
Imediatos
Desvio
Exercícios

São codificadas no formato R e I:

opcode	Rm	shamt	Rn	Rd
11 bits	5 bits	6 bits	5 bits	5 bits

opcode	immediate	Rn	Rd
10 bits	12 bits	5 bits	5 bits

Shifts no formato R, usam *shamt* em vez de *Rm*;

Ex: `LSL X11,X19,#4`

opcode	Rm	shamt	Rn	Rd
1691	0	4	19	11

Imediatos Longos

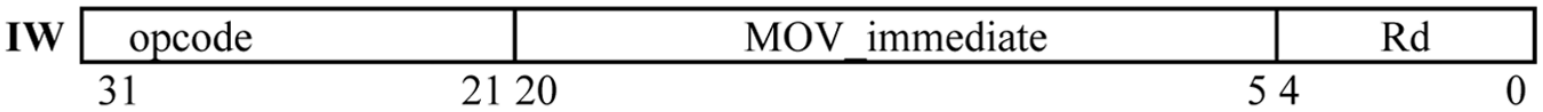
Codificação
Lógicas
Imediatos
Desvio
Exercícios

Em muitas ocasiões, o espaço disponível na instrução para o imediato é pequeno (12 bits). Para resolver o problema, temos duas instruções capazes de imediatos longos: `MOVZ` e `MOVK`.

- Ambas colocam um imediato de 16 bits um registrador, em um quadrante específico.
- `MOVZ` preenche os bits restantes com zero, `MOVK` não os altera.

Exemplo: `MOVZ X9,255, LSL 16` fará com que o registrador `X9` tenha 0x00FF no segundo quadrante:
`X9=0x0000 0000 00FF 0000`

Formato IW



- *opcode:* o código que representa a instrução e o quadrante.
- *MOV_immediate:* Imediato.
- *Rd:* O destino do resultado da operação.

Por exemplo, `MOVK X9, 255, LSL 16`

110100101	01	0000 0000 1111 1111	01001

Da calculadora para o processador

Codificação
Lógicas
Imediatos
Desvio
Exercícios

- O que temos até agora é uma calculadora com memória.
  - Podemos dar uma lista de instruções a nossa calculadora, mas ainda assim ela não seria **programável**.
- O que falta para termos um processador?
- Para as instruções a serem executadas dependerem das entradas, usamos comandos como **IF** e **FOR**.
- Instruções de **desvio** é do que precisamos!

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Desvios

Desvios

Codificação
Lógicas
Imediatos
Desvio
Exercícios

As instruções abaixo podem desviar o programa:

```
1   CBZ   reg , L1
2   CBNZ  reg , L1
3   B     L1
```

- CBZ: *Compare and Branch if Zero*
- CBNZ: *Compare and Branch if Not Zero*
- B: *Branch*

Execução Condicional

Codificação
Lógicas
Imediatos
Desvio
Exercícios

```
1  if (i == j)
2    f = g + h;
3  else
4    f = g - h;
```

```
1          SUB X9,X22,X23   // X9 = i - j
2          CBNZ X9,Else     // vai para Else se i!=j
3                           // (X9!=0)
4          ADD X19,X20,X21  // f=g+h (pula se i!=j)
5          B Exit           // vai para Exit
6  Else:   SUB X19,X20,X21  // f=g-h (pula se i=j)
7  Exit:
```

Laços

Codificação
Lógicas
Imediatos
Desvio
Exercícios

```
1  while (save[i]==k)
2    i+=1;
```

```
1  Loop: LSL X10,X22,#3      // X10=i*8
2        ADD X10,X10,X25     // X10=end de save[i]
3        LDUR X9, [X10,#0]   // X9=save[i]
4        SUB X11,X9,X24      // X11=save[i]-k
5        CBNZ X11, Exit      // vai para Exit se
6                            // save[i]!=k (X11!=0)
7        ADDI X22,X22,#1     // i=i+1
8        B Loop              // vai para Loop
9  Exit:
```

Outros Desvios

CBZ, CBNZ e B resolvem boa parte dos desvios, mas temos outros baseados no B.cond. Os *flags* que existem no LEGv8 são:

- N: se o resultado é negativo
- Z: se o resultado é zero (idem CBZ/CBNZ)
- V: se o resultado causou *overflow*
- C: se o resultado causou *carry*

Para isso, existe a instrução `B.cond`, onde `cond` é um mnemônico que especifica o conjunto de *flags* que devem ou não acontecer para que o desvio ocorra.
Exemplo: `B.VS Label` desvia se houve *overflow*.

Outros Desvios

flags de A - B permitem comparar A e B:

	Signed numbers		**Unsigned numbers**	
Comparison	Instruction	CC Test	Instruction	CC Test
=	B.EQ	Z=1	B.EQ	Z=1
≠	B.NE	Z=0	B.NE	Z=0
<	B.LT	N!=V	B.LO	C=0
≤	B.LE	~(Z=0 & N=V)	B.LS	~(Z=0 & C=1)
>	B.GT	(Z=0 & N=V)	B.HI	(Z=0 & C=1)
≥	B.GE	N=V	B.HS	C=1

Para atualizar *flags*, há instruções lógicas/aritméticas específicas:

```
1  SUBS X9,X10,X11
2  B.LE Label          // desvia se X10 <= X11
```

Outros Desvios

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Por último, temos o desvio para o endereço de um registrador:

```
1   BR  X9
```

Este desvio incondicional vai diretamente para o endereço de memória contido no registrador.

Exercício 1

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Gere o código de máquina correspondente ao código C abaixo. Assuma que *i* está em X9.

```
1    i=i+1;
```

Exercício 2

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Considerando que o processador LEGv8 começa executando a instrução na posição zero da memória, qual o conteúdo da memória correspondente ao seguinte trecho? Dica: use hexadecimal e considere o alinhamento correto para a as instruções.

```
1   SUB   X22,X22,X22
2   ADDI  X22,X22,#16
3   LDUR  X9,[X22,#0]
4   SUBI  X10,X22,#4
```

Exercício 3

Codificação
Lógicas
Imediatos
Desvio
Exercícios

Gere o código assembly correspondente ao código C abaixo:

```
1     temp = divisor;
2     quociente = 0;
3     while(dividendo >= temp){
4         quociente = quociente + 1;
5         temp = temp + divisor;
6     }
```

Obrigado!


Universidade de São Paulo
Departamento de Engenharia de Computação e Sistemas Digitais
PCS

Referências


D. Patterson and J. Hennessy.
Computer Organization and Design ARM Edition: The Hardware Software Interface.
The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Science, 2016.