Arquitetura ARC

Disciplina: Introdução à Arquitetura de Computadores

Luciano Moraes Da Luz Brum

Universidade Federal do Pampa – Unipampa – Campus Bagé

Email: lucianobrum18@gmail.com

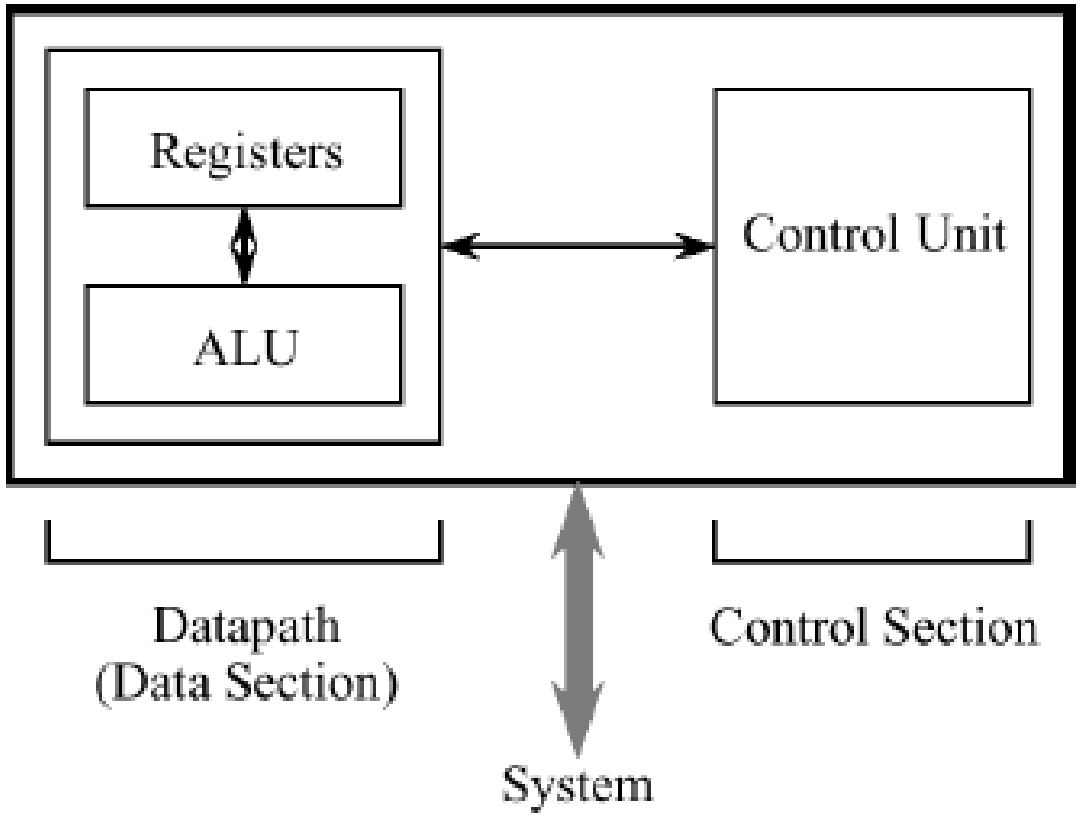
Roteiro

- **Arquitetura ARC – Introdução;**
- **Arquitetura ARC – Memória;**
- **Arquitetura ARC – Registradores;**
- **Arquitetura ARC – ISA;**
- **Arquitetura ARC – Organização e Caminho de Dados;**
- **Arquitetura ARC – Montagem;**
- **Resumo;**

- ARC, um exemplo de arquitetura RISC:
  - ARC é uma máquina de 32 bits com memória endereçada por byte;
  - Podem manipular dados de 32 bits;
  - Tem áreas de memórias separadas por função;

- A CPU consiste de um bloco contendo:
  - Registradores;
  - Unidade Lógica e Aritmética;
  - Unidade de Controle;

- O bloco ou a seção de dados também é conhecida como Datapath (Caminho de dados).



- A unidade de controle é responsável pela busca das instruções da memória e por comandar a execução das instruções buscadas da memória.

- Utiliza dois registradores especiais, que são a interface entre a unidade de controle e de dados. São o IR e PC, o primeiro contém a instrução lida da memória. O segundo possui o endereço de memória a ser lida a próxima instrução.

Ciclo de Busca – Decodificação - Execução

- Algoritmo básico (ciclo de busca e execução):

1. Ler a próxima instrução, apontada pelo PC;
2. Decodificar os operandos;
3. Ler o operando da memória, se houver;
4. Executar a instrução e armazenar os resultados;
5. Repetir o passo 1;

Roteiro

- ~~Arquitetura ARC – Introdução;~~
- **Arquitetura ARC – Memória;**
- **Arquitetura ARC – Registradores;**
- **Arquitetura ARC – ISA;**
- **Arquitetura ARC – Organização e Caminho de Dados;**
- **Arquitetura ARC – Montagem;**
- **Resumo;**

- Os menores 2048 bits são reservados para o S.O.
- Acima de 2048 e abaixo da pilha é reservado para os programas do usuário.
- A pilha está localizada em 2^{31}- 4 e cresce em direção a endereços menores.
- Entre 2^{31} e $2^{32} - 1$ é reservado para dispositivos de entradas e saídas.

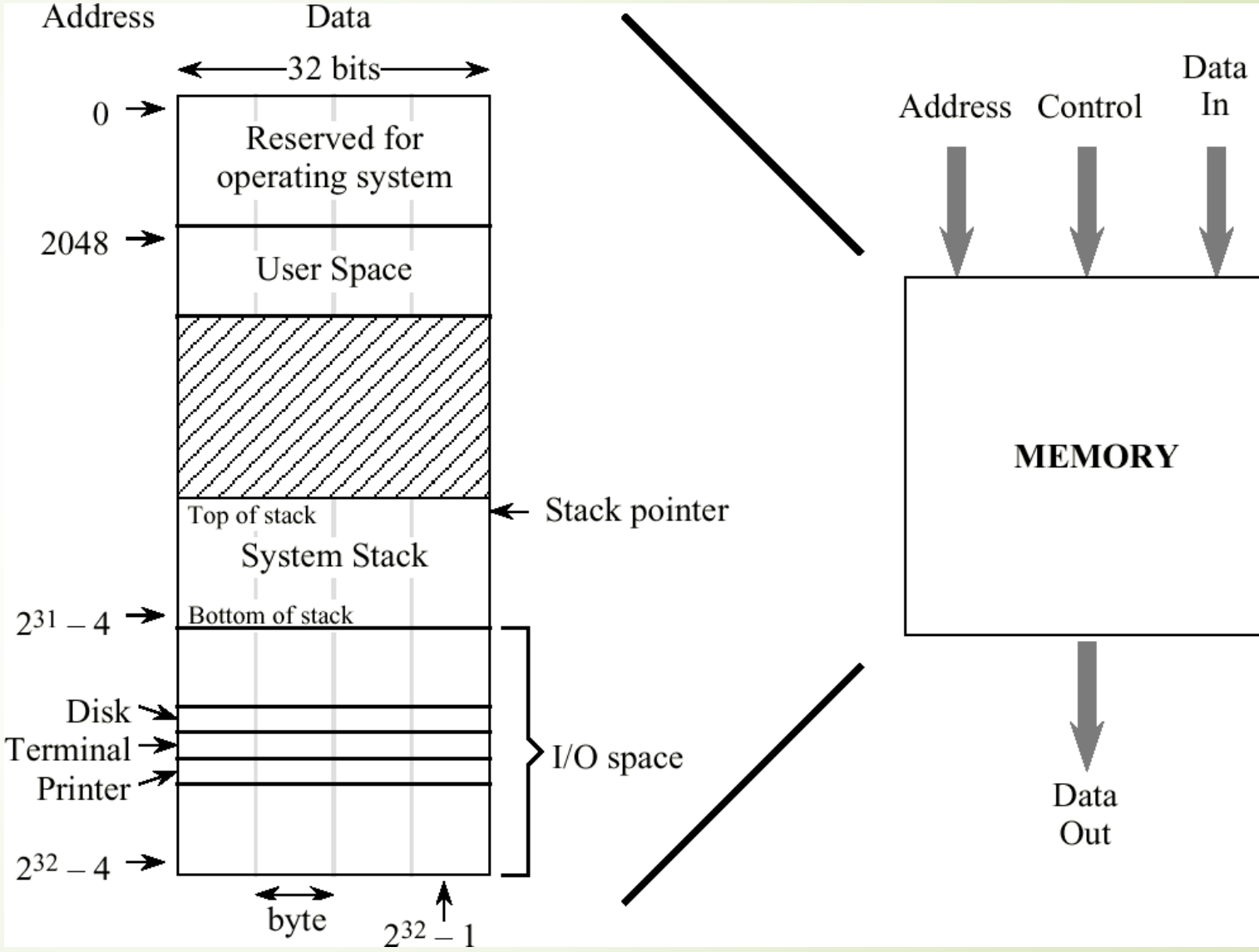


Figura 1: Memória da arquitetura ARC. Fonte: Murdocca e Heuring, 2007.

Roteiro

- ~~Arquitetura ARC – Introdução;~~
- ~~Arquitetura ARC – Memória;~~
- **Arquitetura ARC – Registradores;**
- **Arquitetura ARC – ISA;**
- **Arquitetura ARC – Organização e Caminho de Dados;**
- **Arquitetura ARC – Montagem;**
- **Resumo;**

- Registradores:
  - Os Registradores são memórias especiais e são diferentes da memória, por estarem internas a CPU.
  - O ARC tem 32 registradores de 32 bits;
  - Possui alguns registradores específicos: PC, IR e PSR;
    - PC - Program Count;
    - IR - Instruction Register;
    - PSR – registrador de status;
  - Load-Store – é uma arquitetura do tipo “carrega e armazena”. Todos dados são carregados em registradores para processamentos. O resultado é também sempre armazenado em um registrador (exceto em instruções de leitura e escrita em memória).

- Registrador PSR: Especifica o status do resultado de uma operação aritmética.
  - **“z” → Operação resultou em zero;**
  - **“c” → Operação resultou em carry;**
  - **“n” → Resultado negativo;**
  - **“v” → Resultado da operação gerou um estouro de representação;**
- Registrador PC: Contém o endereço da próxima instrução a ser executada.
- Registrador %r14: Ponteiro do topo da pilha (stack pointer).
- Registrador %r15: Contém endereço de retorno de procedimentos/funções.
- Registrador %r0: Contém a constante 0.

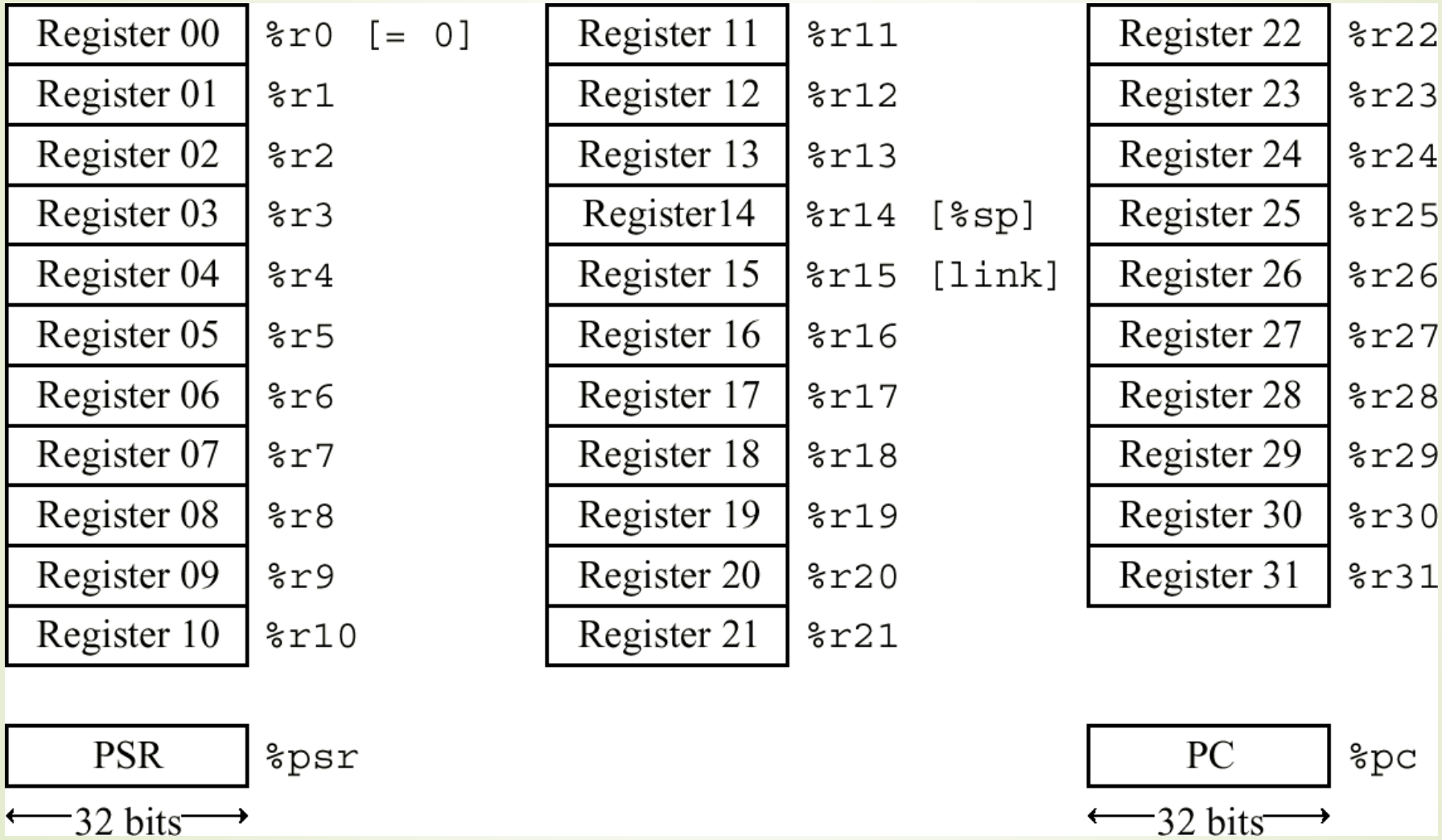


Figura 2: Conjunto de registradores da arquitetura ARC. Fonte: Murdocca e Heuring, 2007.

- Algumas arquiteturas possuem outros registradores especiais:
  - Registradores de indexação de memória, para marcar o endereço inicial e final de um vetor;
  - Registradores de ponto flutuante: muitos processadores utilizam esses registradores, que tratam com operandos em ponto flutuante;
  - Registradores de tempo: registradores com alta precisão que registram o clock da máquina (armazenam o tempo com uma alta precisão);
  - Registradores de apoio ao Sistema Operacional;
  - Registradores de uso do Sistema Operacional;

Roteiro

- ~~Arquitetura ARC – Introdução;~~
- ~~Arquitetura ARC – Memória;~~
- ~~Arquitetura ARC – Registradores;~~
- **Arquitetura ARC – ISA;**
- **Arquitetura ARC – Organização e Caminho de Dados;**
- **Arquitetura ARC – Montagem;**
- **Resumo;**

- Instruction Set Architecture (Conjunto de instruções da arquitetura):

  - São uma coleção de instruções/comandos que são usados para executar tarefas de receber, processar e transmitir dados dentro do processador.

  - Cada arquitetura pode ter diferentes conjuntos de instruções.

  - Necessidades de compiladores: C, Pascal, Java, etc.

Compiladores

- Software utilizado para gerar programas em linguagem de máquinas.

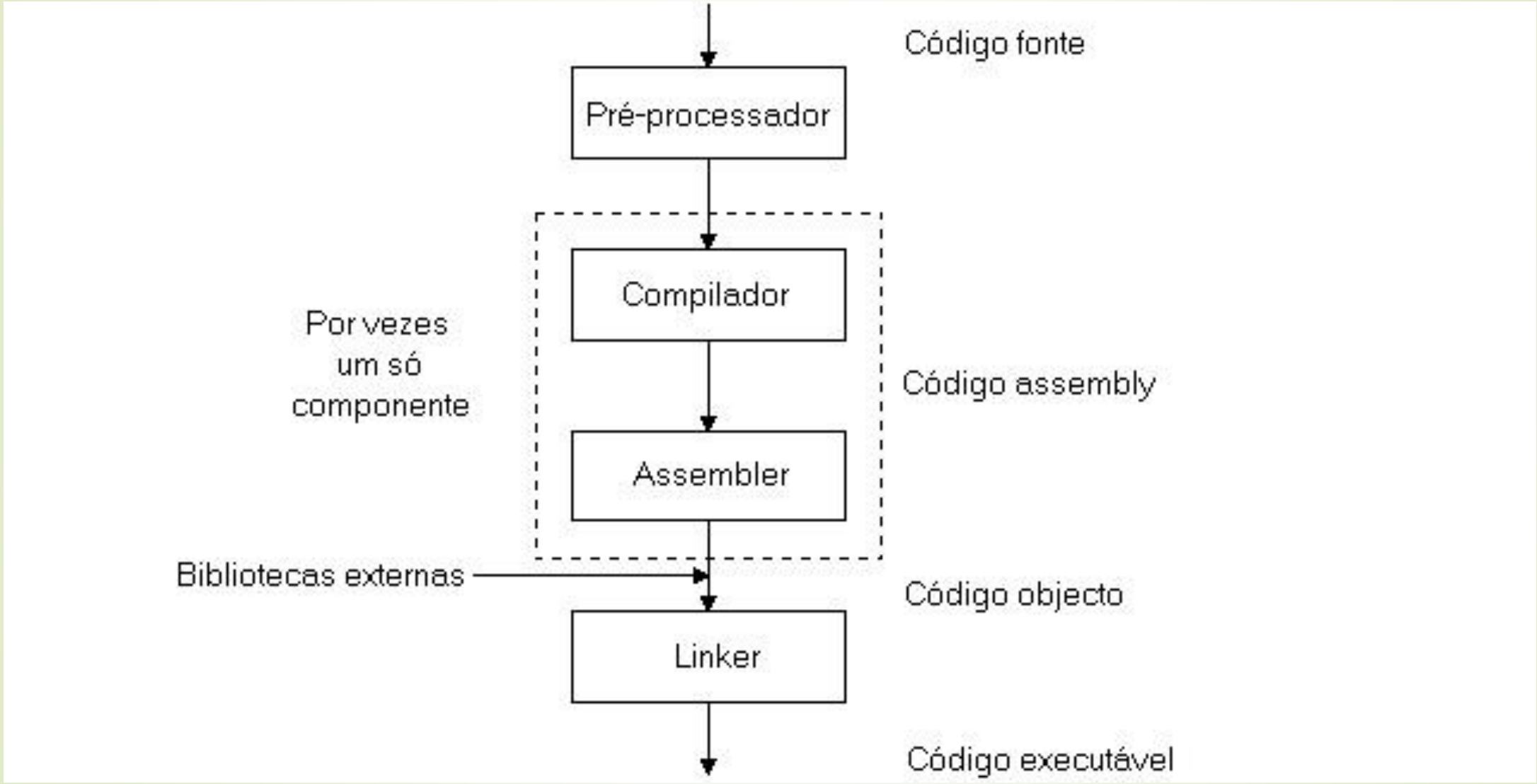
  Como entrada, são usados linguagens de alto nível, como a linguagem C, Fortran, Pascal, etc.

- Usando o código fonte, por exemplo da linguagem C, é possível compilar um mesmo programa em várias arquiteturas diferentes.

Montadores (assembler)

- Software utilizado para gerar programas em linguagem de máquina. Como entrada, é utilizada a linguagem assembly da arquitetura alvo (Neander, MIPS, ARC, i7, etc)

- É feita a tradução literal das instruções assembly para o código equivalente em linguagem de máquina.


Código fonte
Pré-processador
Compilador
Por vezes um só componente
Código assembly
Assembler
Bibliotecas externas
Código objecto
Linker
Código executável

- Conjunto de Instruções ARC (15 instruções):
  - Movimentação de dados:
    - ld → Leitura de dados da memória para o registrador;
    - st → Escrita de dados do registrador para a memória;
  - Instruções de Controle – instruções que causam desvio no fluxo de execução:
    - Desvio incondicional: call e jmpl;
    - Desvio condicional: be, bneg, bcs, bvs, ba (branchs);
  - Instrução Aritmética:
    - addcc → instrução de soma de operandos;
  - Instruções Lógicas: instruções de operações lógicas:
    - andcc, orcc, orncc, srl, sethi;

	Mnemonic	Meaning
Memory	ld	Load a register from memory
	st	Store a register into memory
Logic	sethi	Load the 22 most significant bits of a register
	andcc	Bitwise logical AND
	orcc	Bitwise logical OR
	orncc	Bitwise logical NOR
	srl	Shift right (logical)
Arithmetic	addcc	Add
Control	call	Call subroutine
	jmpl	Jump and link (return from subroutine call)
	be	Branch if equal
	bneg	Branch if negative
	bcs	Branch on carry
	bvs	Branch on overflow
	ba	Branch always

Figura 3: Conjunto de instruções da arquitetura ARC. Fonte: Murdocca e Heuring, 2007.

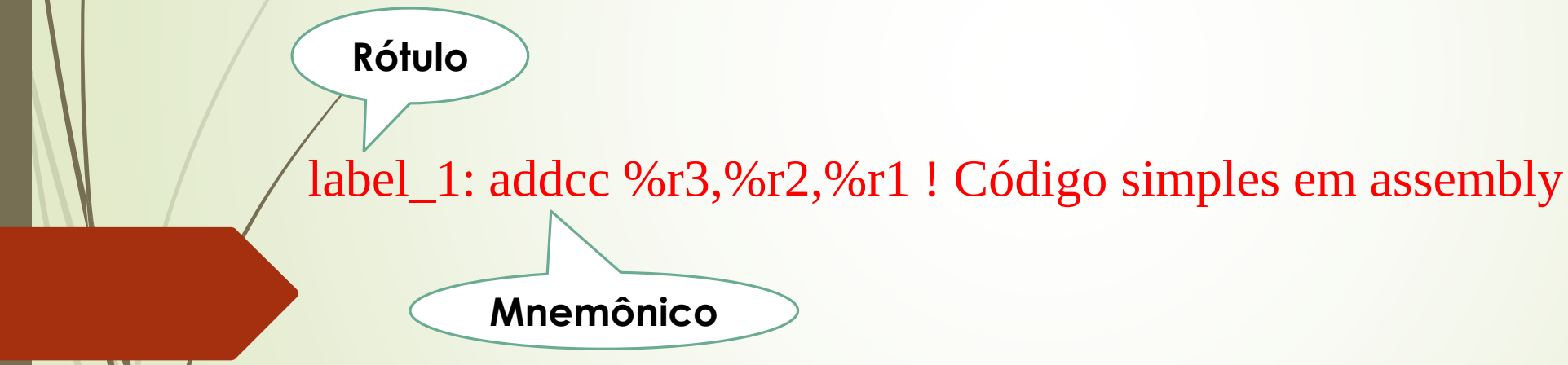
- Formato das instruções ARC:

label_1: addcc %r3,%r2,%r1 ! Código simples em assembly

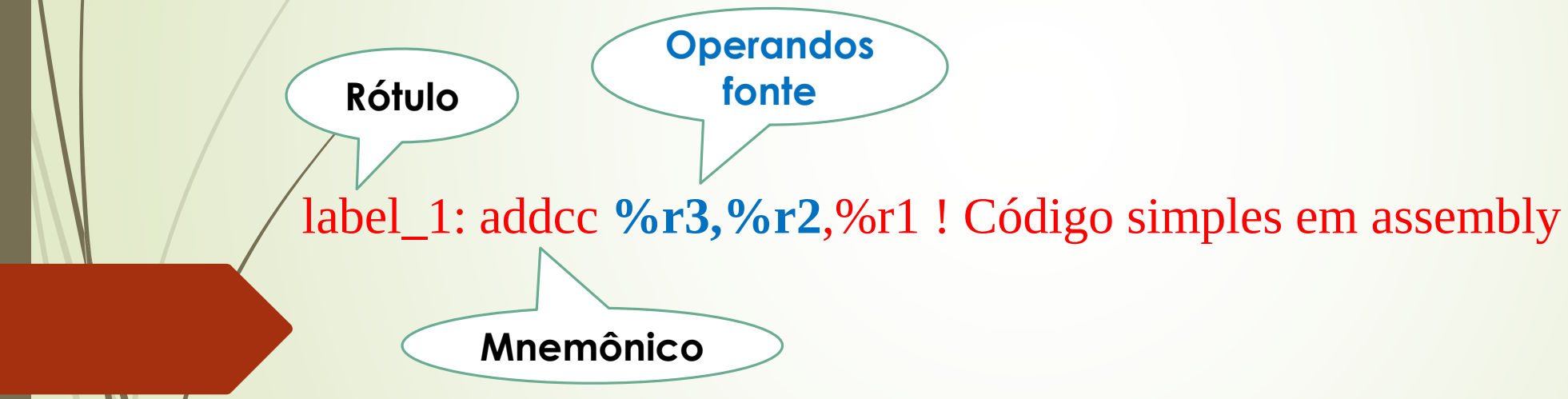
- Formato das instruções ARC:

label_1: addcc %r3,%r2,%r1 ! Código simples em assembly

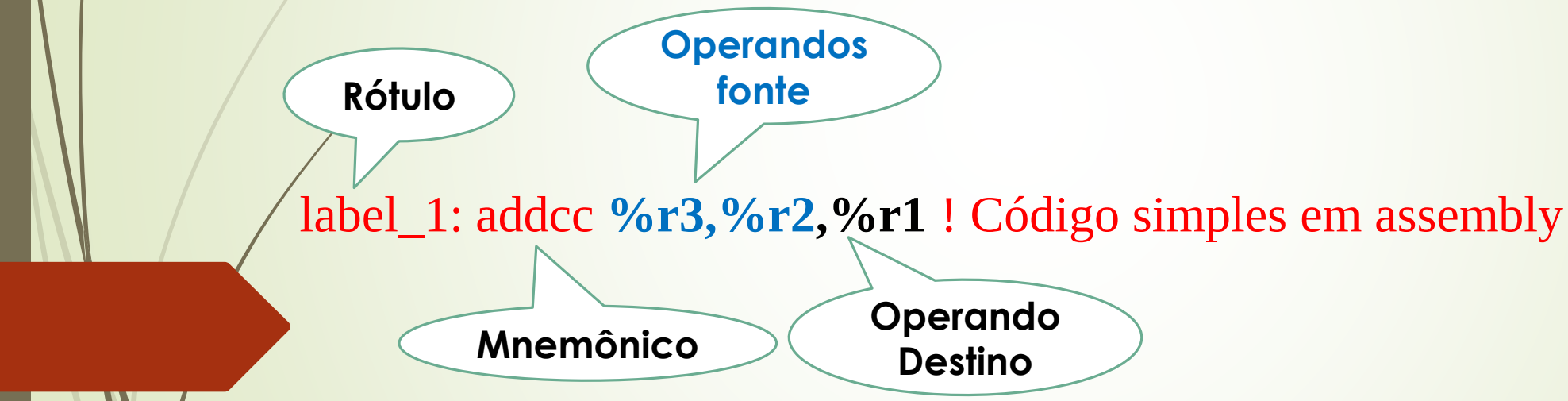
- Formato das instruções ARC:



- Formato das instruções ARC:



- Formato das instruções ARC:



- Formato das instruções ARC:

ARC Pseudo-Ops

Pseudo-Op	Usage	Meaning
`.equ X`	`.equ #10`	Treat symbol `X` as $(10)_{16}$
`.begin`	`.begin`	Start assembling
`.end`	`.end`	Stop assembling
`.org`	`.org 2048`	Change location counter to 2048
`.dwb`	`.dwb 25`	Reserve a block of 25 words
`.global`	`.global Y`	`Y` is used in another module
`.extern`	`.extern Z`	`Z` is defined in another module
`.macro`	`.macro M a, b, ...`	Define macro `M` with formal parameters `a, b, ...`
`.endmacro`	`.endmacro`	End of macro definition
`.if`	`.if <cond>`	Assemble if `<cond>` is true
`.endif`	`.endif`	End of `.if` construct

- Obs: Não fazem parte da ISA, e sim do assembler (montador);

- Exemplo de programa em linguagem assembly:

```
! This programs adds two numbers
        .begin
        .org 2048
prog1:  ld      [x], %r1            ! Load x into %r1
        ld      [y], %r2            ! Load y into %r2
        addcc   %r1, %r2, %r3       ! %r3 ← %r1 + %r2
        st      %r3, [z]            ! Store %r3 into z
        jmpl    %r15 + 4, %r0       ! Return
x:      15
y:      9
z:      0
        .end
```

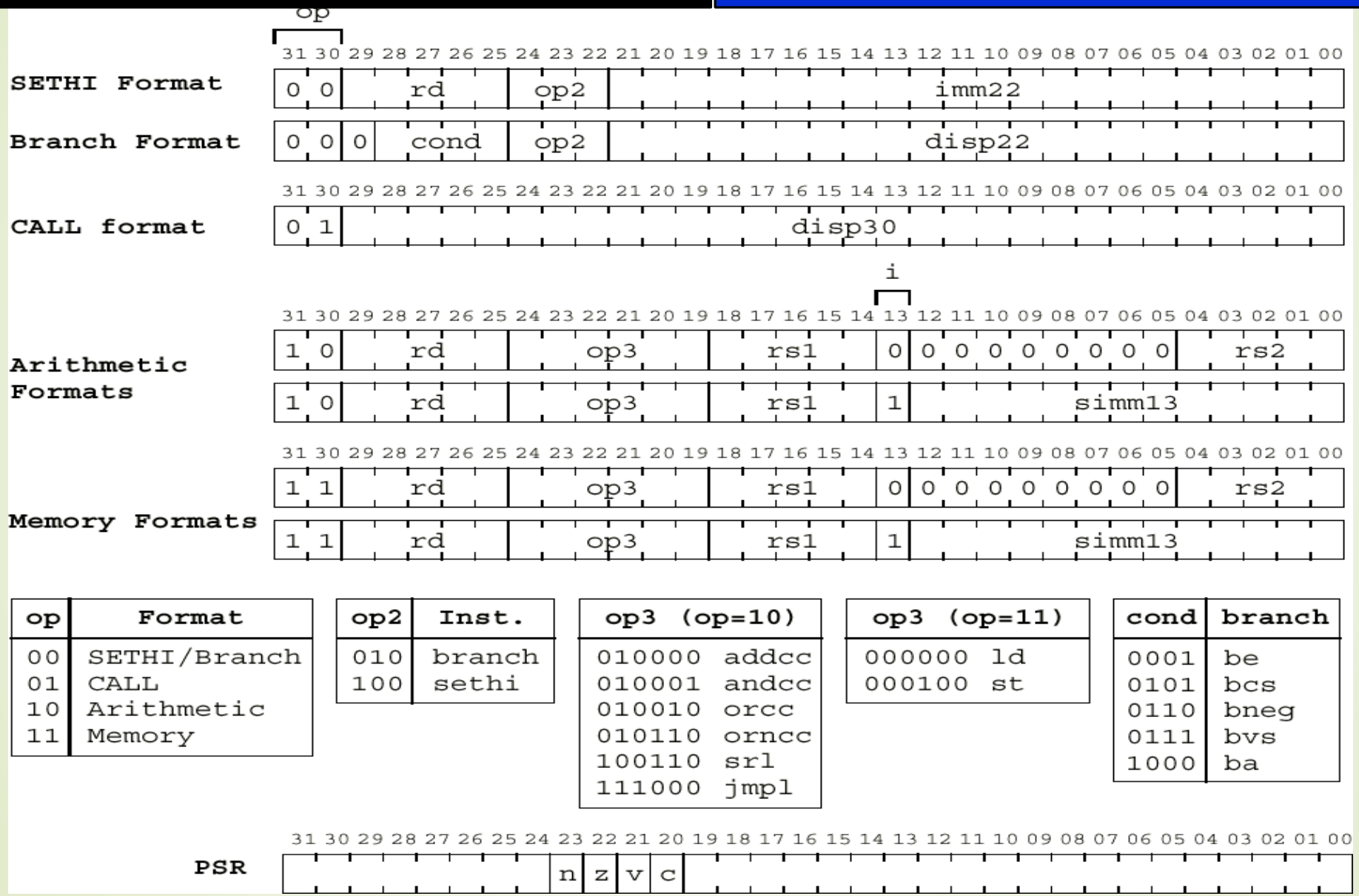


Figura 4: Formato das instruções da arquitetura ARC. Fonte: Murdocca e Heuring, 2007.

Roteiro

- ~~Arquitetura ARC – Introdução;~~
- ~~Arquitetura ARC – Memória;~~
- ~~Arquitetura ARC – Registradores;~~
- ~~Arquitetura ARC – ISA;~~
- **Arquitetura ARC – Organização e Caminho de Dados;**
- **Arquitetura ARC – Montagem;**
- **Resumo;**

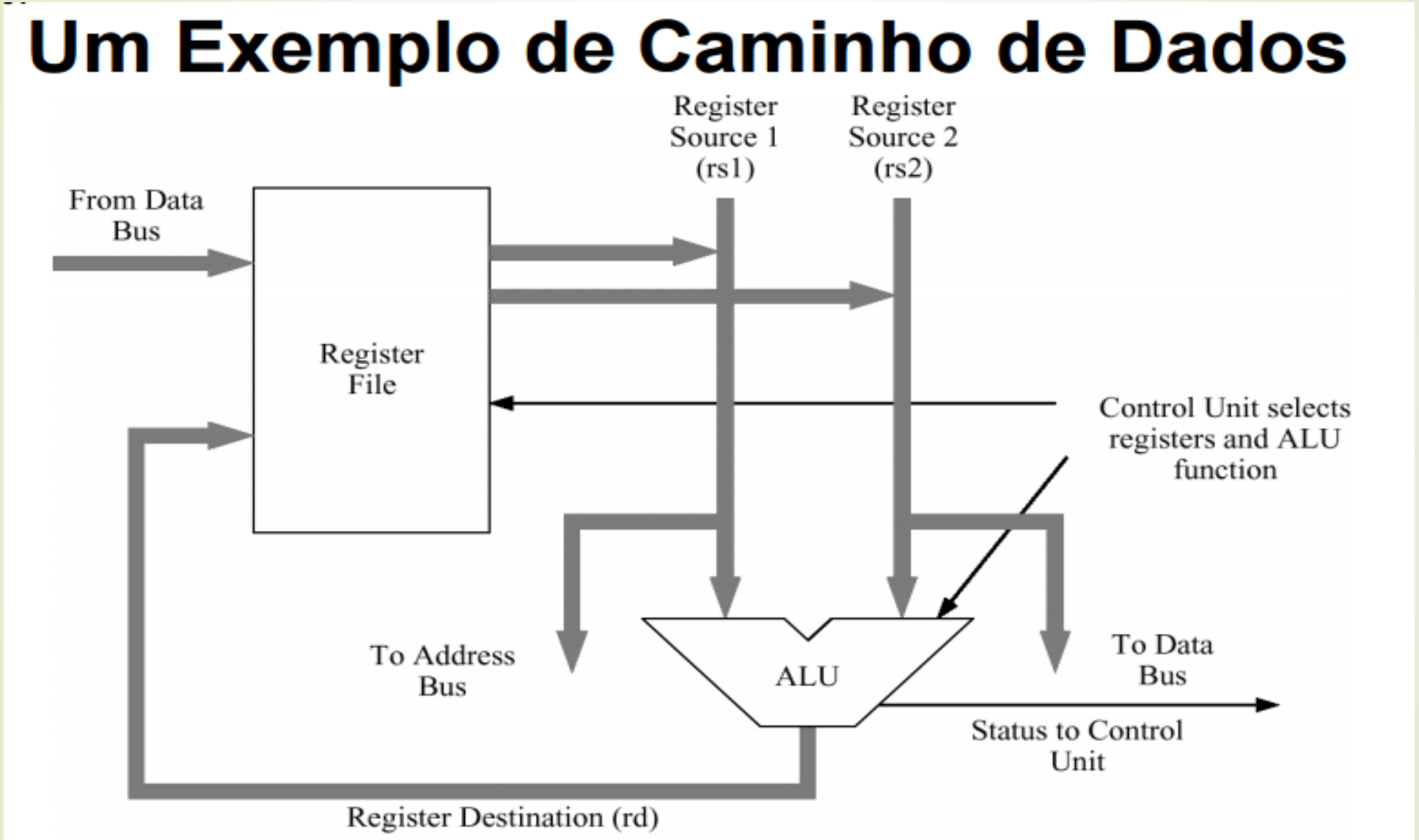


Figura 5: Exemplo de caminho de dados (Datapath) para arquitetura ARC. Fonte: Murdocca e Heuring, 2007.

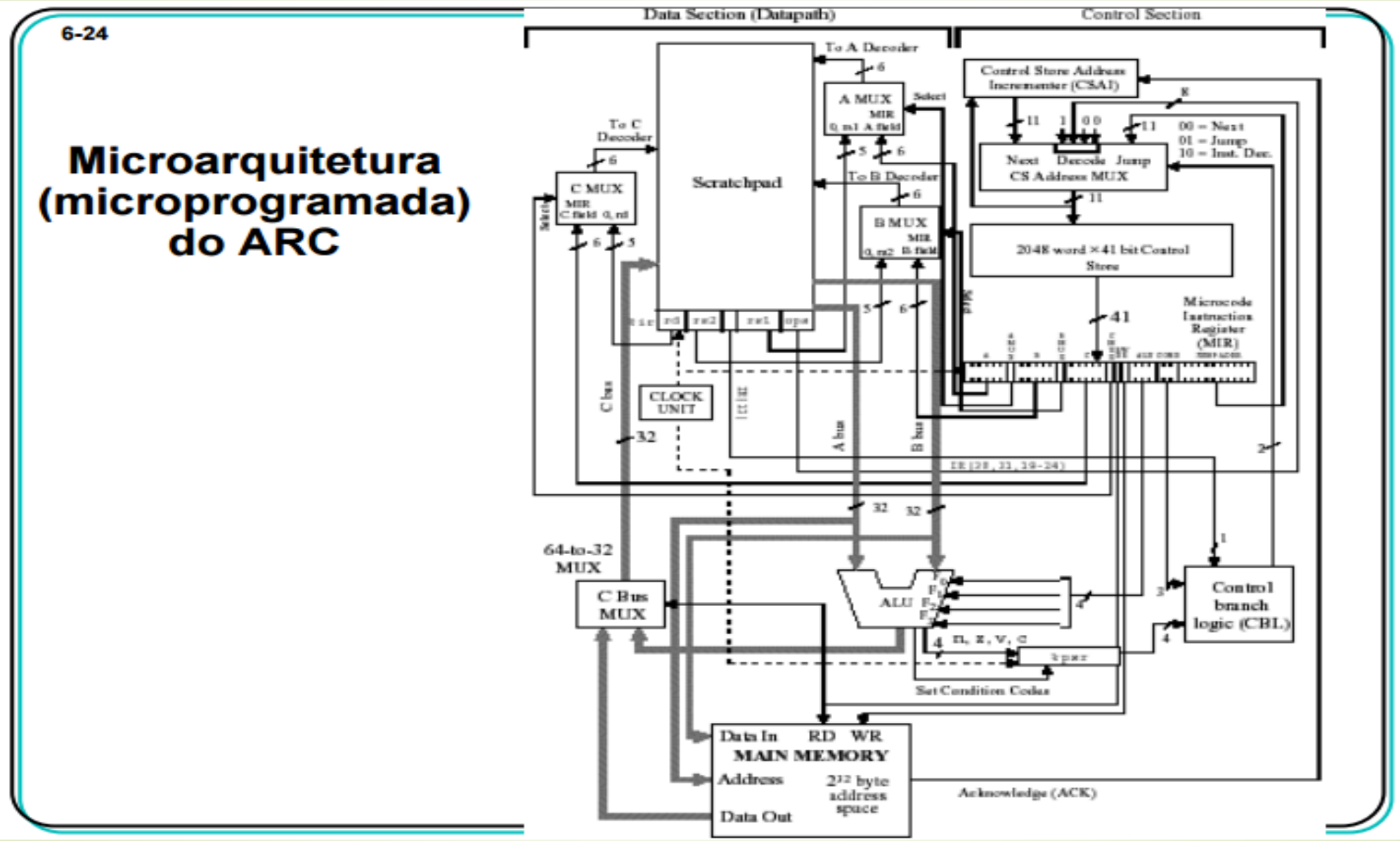


Figura 6: Microarquitetura ARC. Fonte: Murdocca e Heuring, 2007.

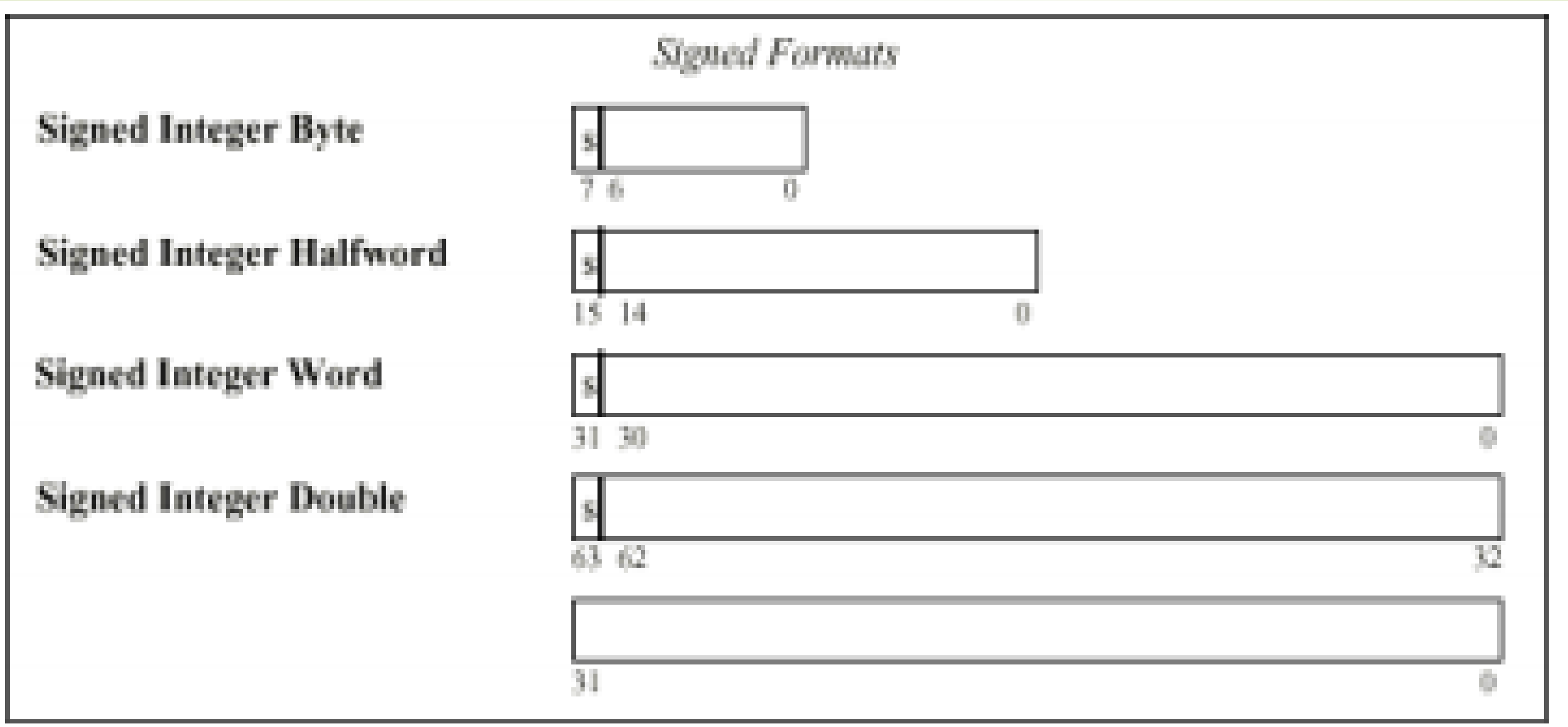


Figura 7: Formato dos dados (com sinal) na arquitetura ARC. Fonte: Murdocca e Heuring, 2007.

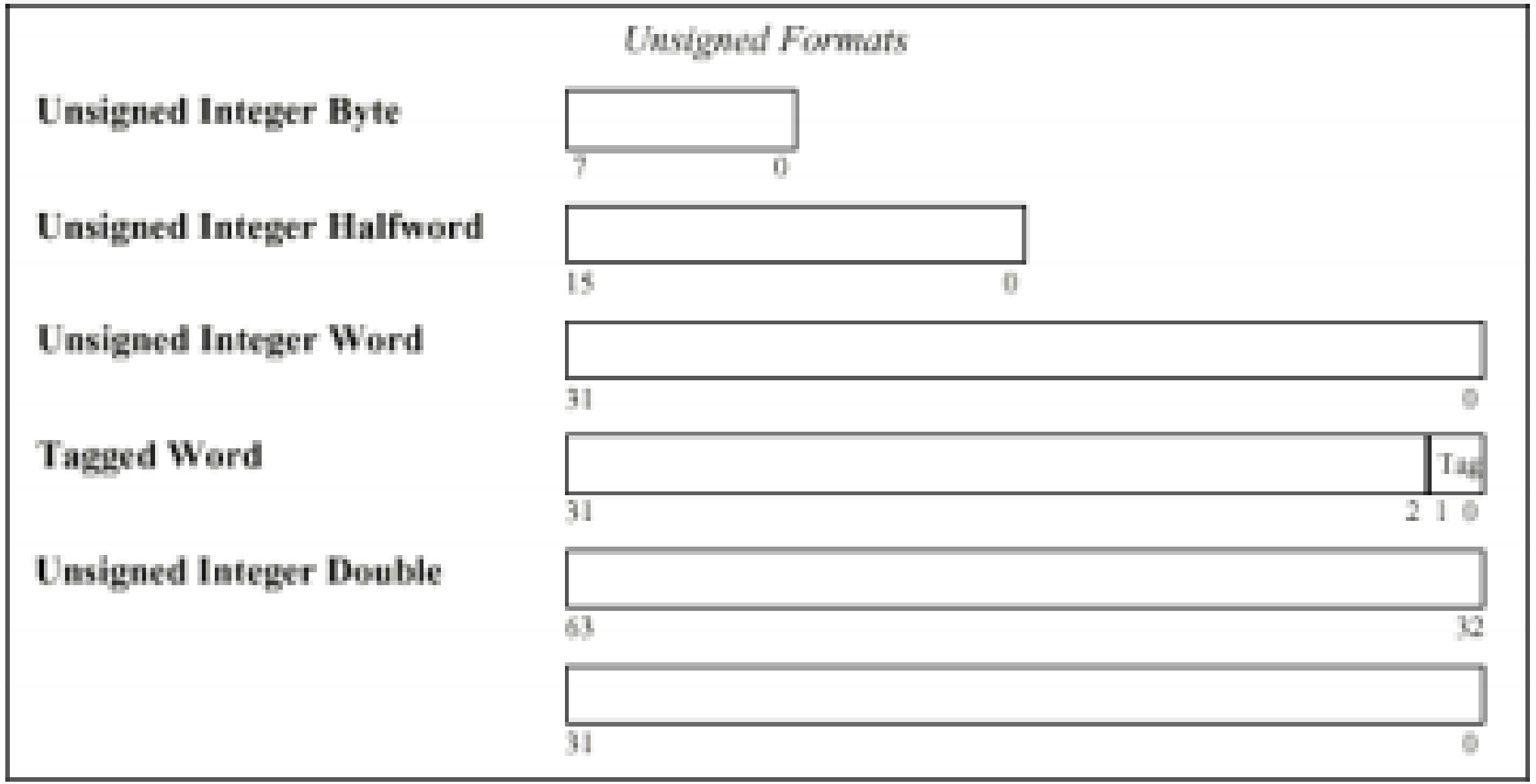


Figura 8: Formato de dados sem sinal na arquitetura ARC. Fonte: Murdocca e Heuring, 2007.

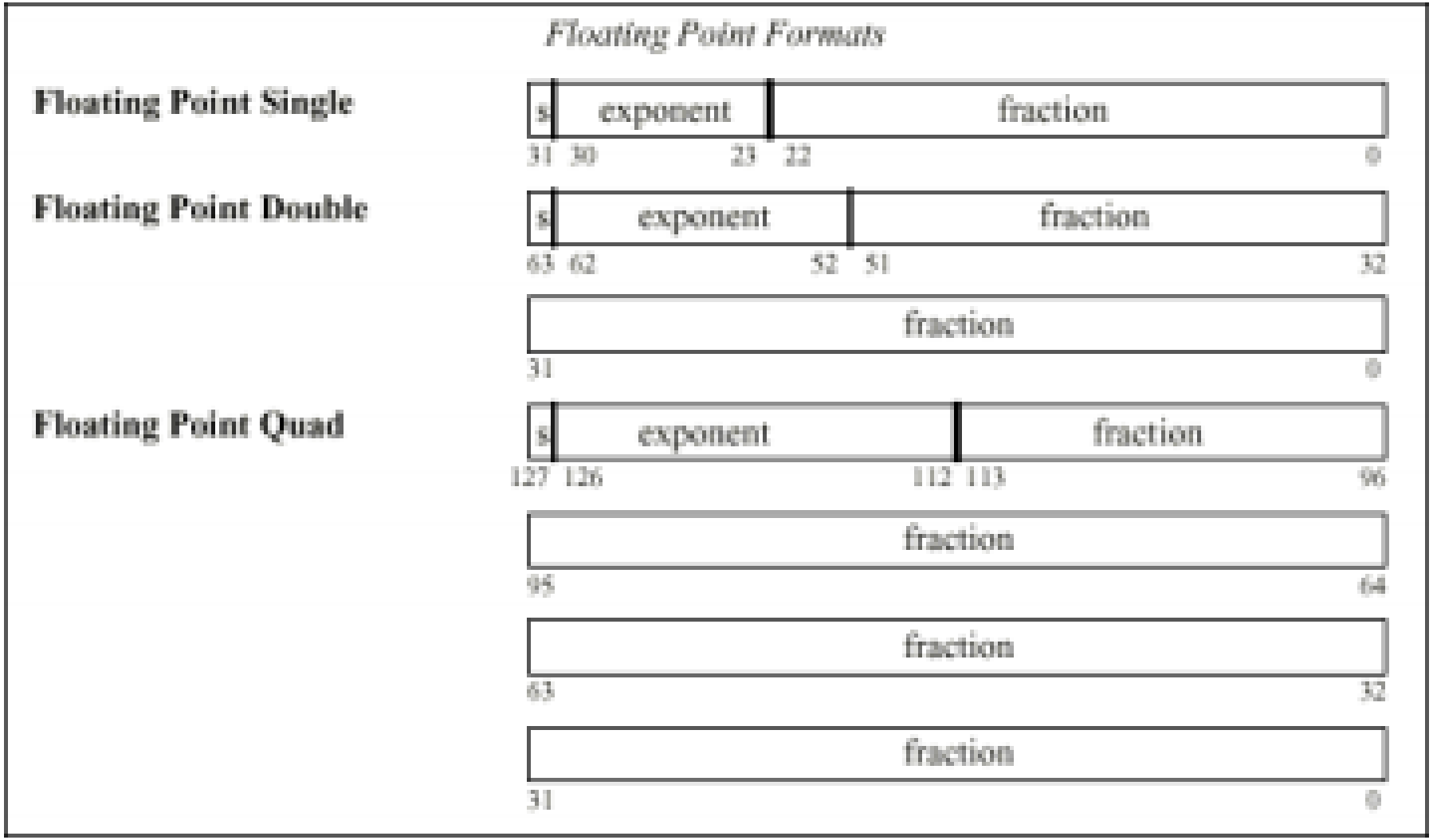


Figura 9: Formato de dados em Ponto Flutuante na arquitetura ARC. Fonte: Murdocca e Heuring, 2007.

Máquinas de 1, 2 e 3 Endereços

- Considere: A = (B*C) + D

- Assumindo:
  - Endereços e dados de 2 bytes;
  - Opcode de 1 byte;
  - Operandos podem ser movidos 2 bytes por vez (de e para memória);

Máquinas de 1, 2 e 3 Endereços

- Considere: A = (B*C) + D
- 3 Endereços:
  - **mult B, C, A**
  - **add D, A, A**
- Tamanho do programa:
- **14 bytes = 12 bytes de operandos e 2 bytes de opcode**
- Tráfego de memória:
- **2 x (7 bytes busca da instrução + 6 bytes busca dos operandos) = 26 bytes**

Máquinas de 1, 2 e 3 Endereços

- Considere: A = (B*C) + D
- 2 Endereços:
  - **load B, A**
  - **mult C, A**
  - **add D, A**
- Tamanho do programa: **15 bytes = 12 bytes de operandos e 3 bytes de opcode**
- Tráfego de memória:
- **3 * (5 bytes busca inst.)** + 2 * (4 bytes busca op. + 2 bytes armazenar op.) + **(2 bytes busca op. + 2 bytes armazenar op.)** = **31 bytes**

Máquinas de 1, 2 e 3 Endereços

- Considere: A = (B*C) + D
- 1 Endereço:
  - **load B**
  - **mult C**
  - **add D**
  - **store A**
- Tamanho do programa: **12 bytes = 8 bytes de operandos e 4 bytes de opcode**
- Tráfego de memória: **4 * (3 bytes busca inst.)** + **6 bytes busca op.** + **2 bytes armazenamento op.** = 20 bytes

Modos de Endereçamento

Modo de endereçamento	Sintaxe	Significado
Imediato	#K	K
Direto	K	M[K]
Indireto	(K)	M[M[K]]
Registrador	Rn	M[Rn]
Registrador Indexado	Rn + Rm	M[Rn + Rm]
Registrador Baseado	Rn + X	M[Rn + X]
Registrador Baseado Indexado	(Rn + Rm + X)	M[Rn + Rm + X]

Figura 10: Modos de endereçamento no ARC. Fonte: Murdocca e Heuring, 2007.

Roteiro

- ~~Arquitetura ARC – Introdução;~~
- ~~Arquitetura ARC – Memória;~~
- ~~Arquitetura ARC – Registradores;~~
- ~~Arquitetura ARC – ISA;~~
- ~~Arquitetura ARC – Organização e Caminho de Dados;~~
- **Arquitetura ARC – Montagem;**
- **Resumo;**

Montagem

- **Relembrando...**
  - **Montagem é o processo de tradução da linguagem assembly para a linguagem de máquina;**
  - **Linguagem de máquina = Binário !!**
  - **Montador/assembler é o processo de tradução automatizado;**

Montagem

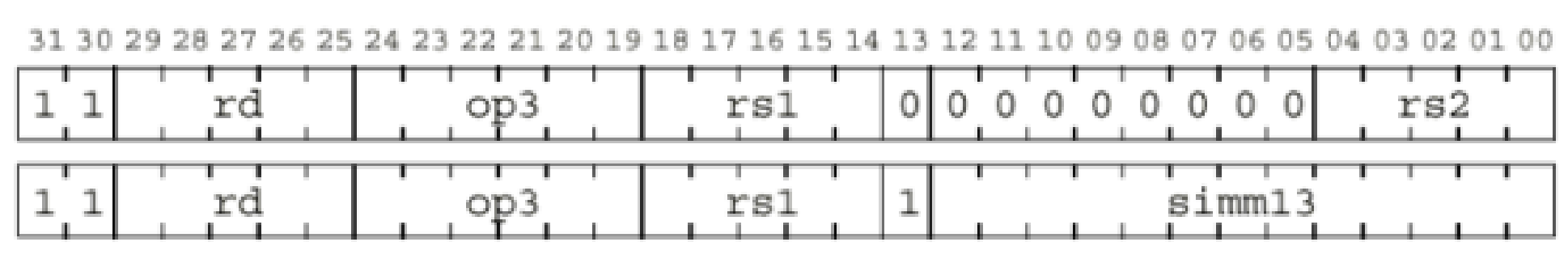
- Montadores de 2 Passos :
  - A maioria dos montadores lê os textos do programa em linguagem de montagem duas vezes:
  - Passo 01: Determina o endereço de todos os itens de dados e instruções. Seleciona quais instruções devem ser geradas para cada instrução em linguagem de montagem (não gera o código). Ao final do primeiro passo todos símbolos estarão identificados e serão armazenados na Tabela de Símbolos.
  - Passo 02: Gera o Código de Máquina.

Montagem

- Montando um código
  - Instrução exemplo:

ld [x], %r1

Reconhecimento de Padrão (Formato de instrução de acesso a memória)



ld [x], %r1

op: ld = 11

rd: %r1 = 00001

op3: op3 = 000000

rs1: %r0 = 00000

i: i = 0 (primeiro formato) 1 (segundo formato)

simm13 = end. rótulo x (2048 + 20 = 2068)

0100000010100

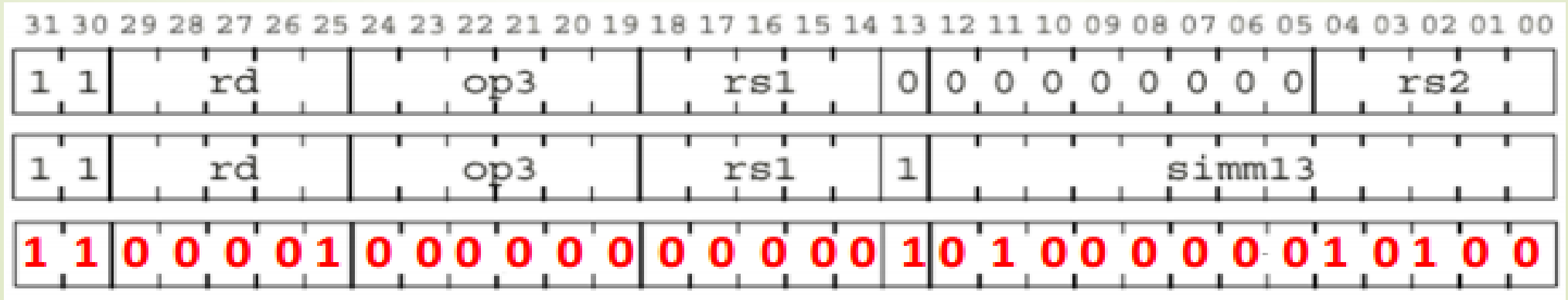
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



ld [y], %r2

op: ld =

rd: %r2 =

op3: op3 =

rs1: %r0 =

i:

simm13 =

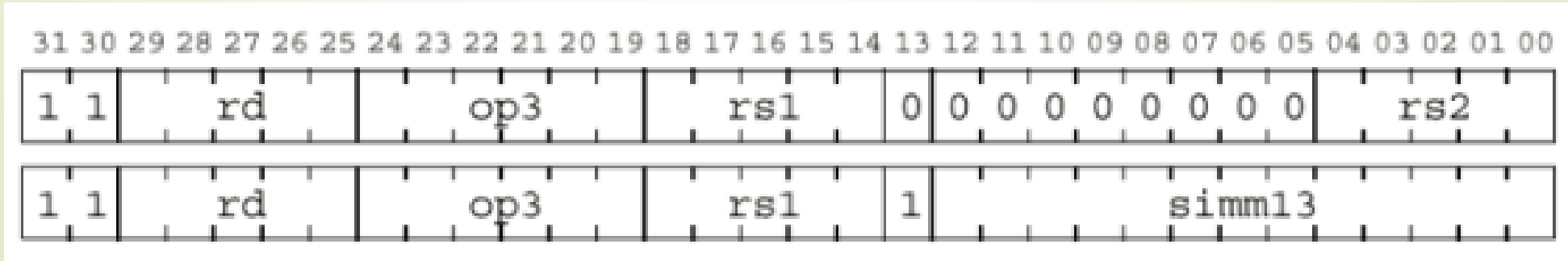
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



ld [y], %r2

op: ld = 11

rd: %r2 = 00010

op3: op3 = 000000

rs1: %r0 = 00000

i: i = 0 (primeiro formato) 1 (segundo formato)

simm13 = end. rótulo x (2048 + 24 = 2072)

0100000011000

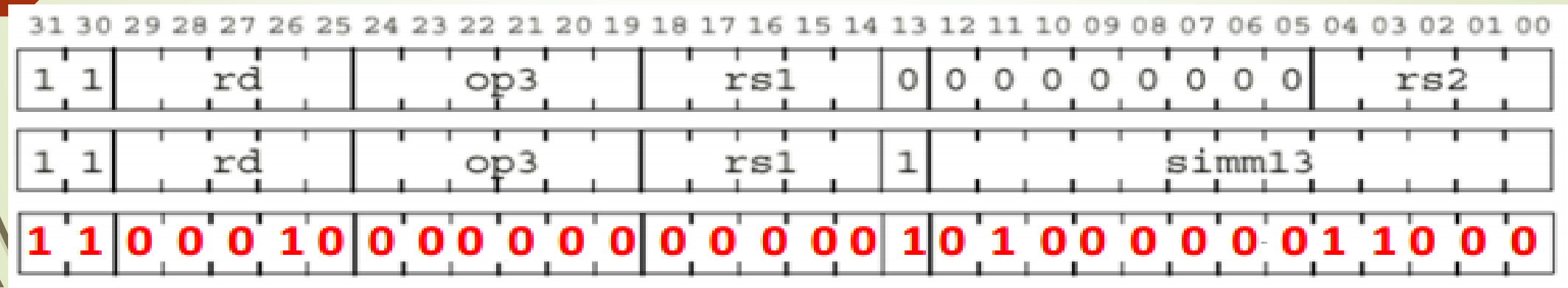
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



addcc %r1, %r2,%r3

op: addcc =

rd: %r3 =

op3: addcc =

rs1: %r1 =

rs2: %r2 =

i:

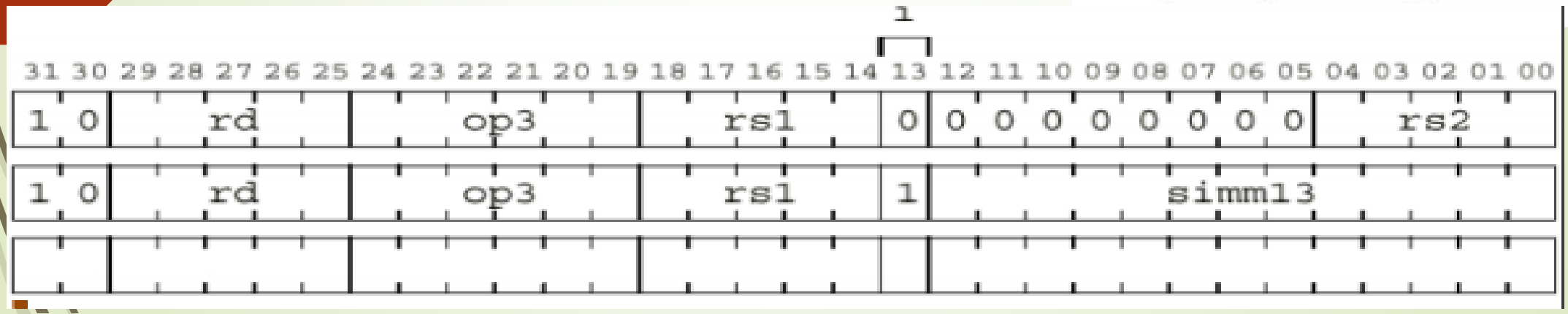
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



addcc %r1, %r2,%r3

op: addcc = 10

rd: %r3 = 00011

op3: addcc = 010000

rs1: %r1 = 00001

rs2: %r2 = 00010

i: 0

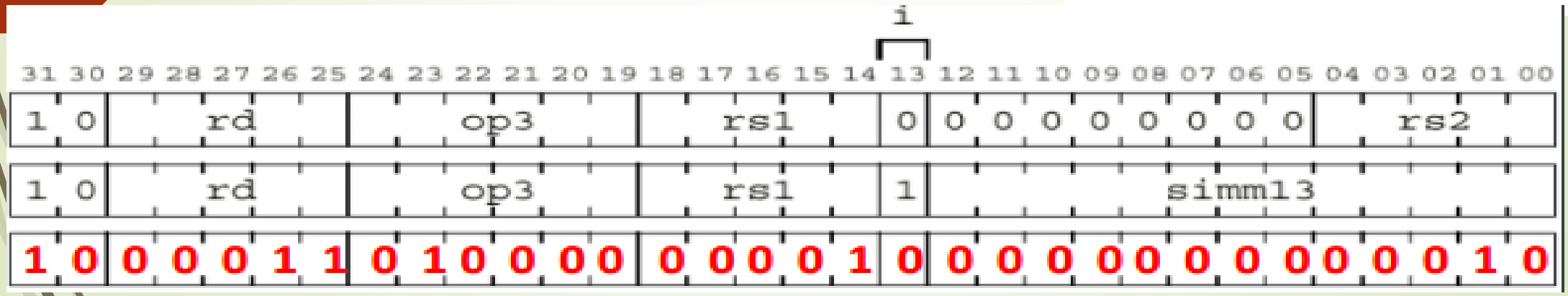
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



st %r3, [z]

op: st=

rd: %r3 =

op3: st =

rs1: %r1 =

simm13:

i:

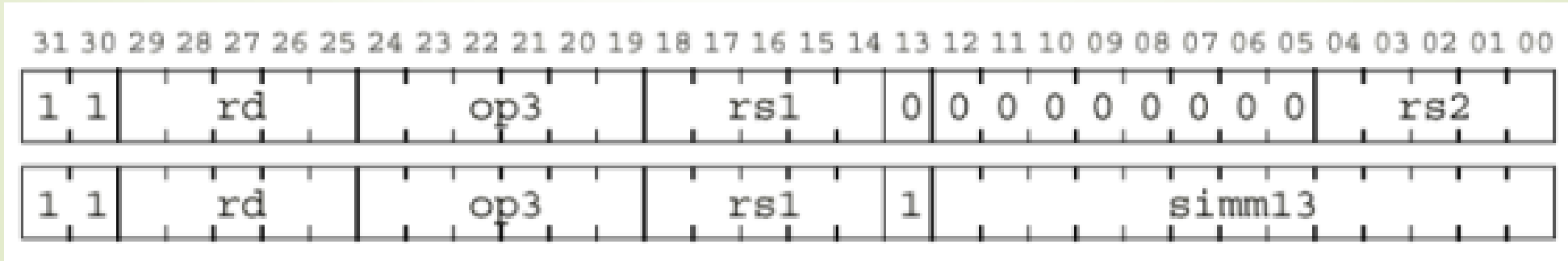
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



st %r3, [z]

op: st= 11

rd: %r3 = 00011

op3: st = 000100

rs1: %r1 = 00000

simm13: 0100000011100

i: 1

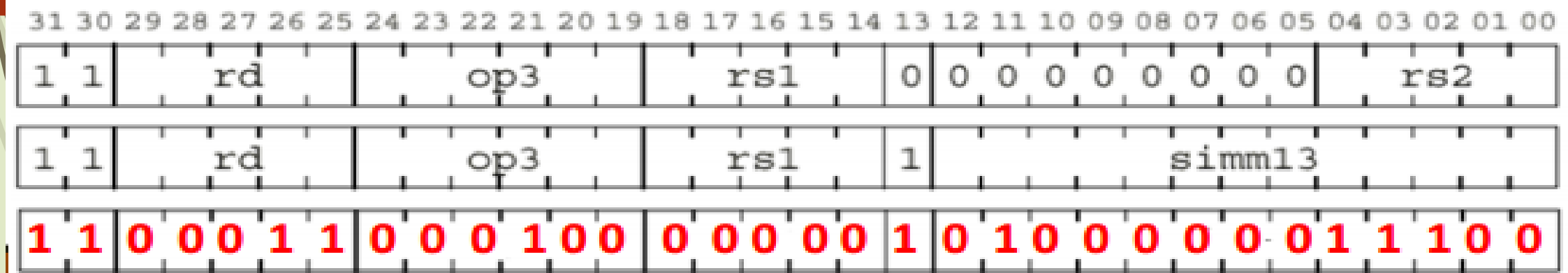
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



jmpl %r15+4, %r0

op: jmpl =

rd: %r0 =

op3: jmpl =

rs1: %r15 =

simm13:

i:

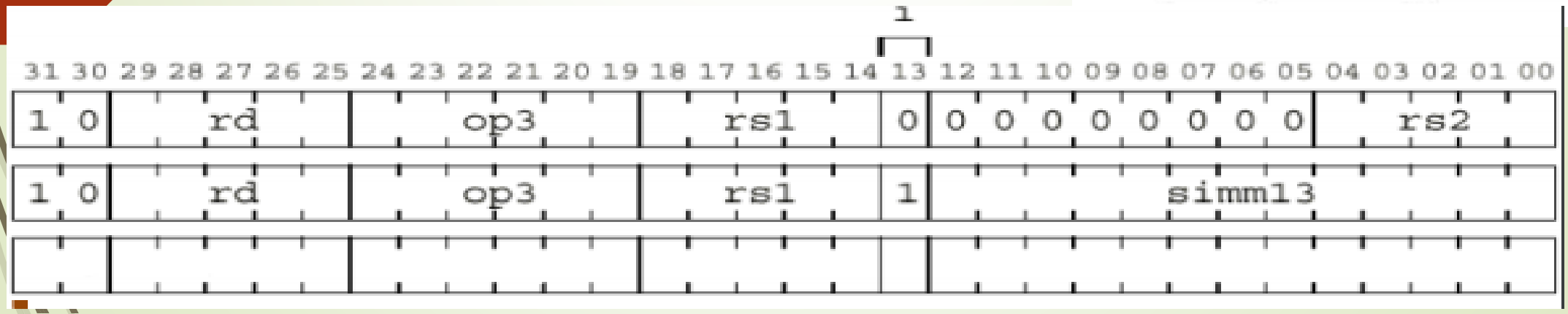
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



jmpl %r15+4, %r0

op: jmpl = 10

rd: %r0 = 00000

op3: jmpl =111000

rs1: %r15 = 01111

simm13: 0000000000100

i: 1

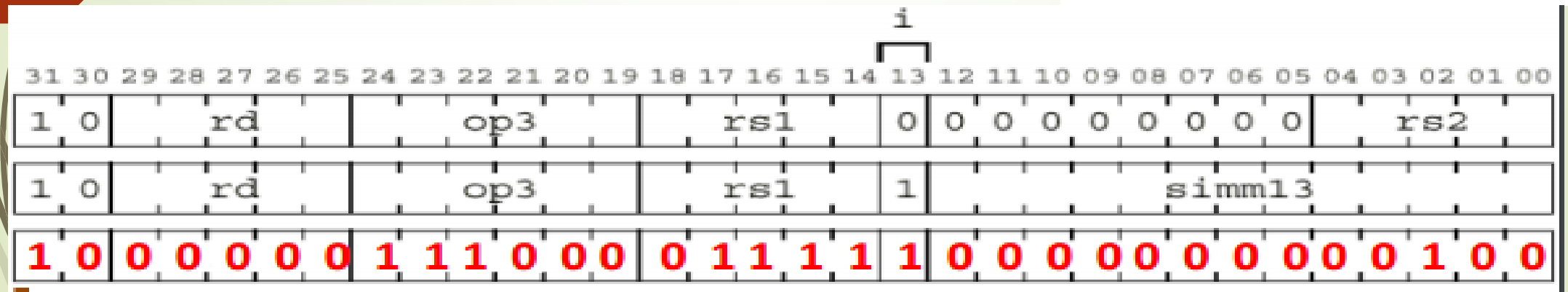
op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st



Resultado da Montagem

Endereço	Instrução	Código de Máquina
	.begin	
	.org 2048	
2048	ld[x], %r1	11000010000000000010100000010100
2052	ld[y], %r2	11000100000000000010100000011000
2056	addcc %r1,%r2,%r3	10000110100000000100000000000010
2060	st %r3, [z]	11000110001000000010100000011100
2064	jmpl %r15+4, %r0	10000001110000111110000000000100
2068	15	00000000000000000000000000001111
2072	9	00000000000000000000000000001001
2076	0	00000000000000000000000000000000

Roteiro

- ~~Arquitetura ARC – Introdução;~~
- ~~Arquitetura ARC – Memória;~~
- ~~Arquitetura ARC – Registradores;~~
- ~~Arquitetura ARC – ISA;~~
- ~~Arquitetura ARC – Organização e Caminho de Dados;~~
- ~~Arquitetura ARC – Montagem;~~
- **Resumo;**

Resumo

- **Características da arquitetura ARC;**
- **Conjunto de registradores e instruções;**
- **Memória;**
- **Formato dos dados;**
- **Linguagem assembly do ARC;**
- **Processo de montagem;**

Dúvidas ?