Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Sistemas e Computação
Curso de Bacharelado em Ciência da Computação

Organização e Arquitetura de Computadores

(Processador – Parte I – B)

Profa. Joseana Macêdo Fechine Régis de Araújo
joseana@computacao.ufcg.edu.br

Carga Horária: 60 horas

Tópicos

- **Organização e Arquitetura Básicas de Computadores**
  - Conceitos Básicos (Processadores) - Assembly

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Exemplos de Instruções

Instruction		Format	Meaning
`add`	`rd, rs1, rs2`	R	Add registers
`sub`	`rd, rs1, rs2`	R	Subtract registers
`sll`	`rd, rs1, rs2`	R	Shift left logical by register
`srl`	`rd, rs1, rs2`	R	Shift right logical by register
`sra`	`rd, rs1, rs2`	R	Shift right arithmetic by register
`and`	`rd, rs1, rs2`	R	Bitwise AND with register
`or`	`rd, rs1, rs2`	R	Bitwise OR with register
`xor`	`rd, rs1, rs2`	R	Bitwise XOR with register
`slt`	`rd, rs1, rs2`	R	Set if less than register, 2's complement
`sltu`	`rd, rs1, rs2`	R	Set if less than register, unsigned
`addi`	`rd, rs1, imm[11:0]`	I	Add immediate
`slli`	`rd, rs1, shamt[4:0]`	I	Shift left logical by immediate
`srli`	`rd, rs1, shamt[4:0]`	I	Shift right logical by immediate
`srai`	`rd, rs1, shamt[4:0]`	I	Shift right arithmetic by immediate
`andi`	`rd, rs1, imm[11:0]`	I	Bitwise AND with immediate
`ori`	`rd, rs1, imm[11:0]`	I	Bitwise OR with immediate
`xori`	`rd, rs1, imm[11:0]`	I	Bitwise XOR with immediate
`slti`	`rd, rs1, imm[11:0]`	I	Set if less than immediate, 2's complement
`sltiu`	`rd, rs1, imm[11:0]`	I	Set if less than immediate, unsigned
`lui`	`rd, imm[31:12]`	U	Load upper immediate
`auipc`	`rd, imm[31:12]`	U	Add upper immediate to pc

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Instruções (Acesso à Memória)

Instruction	Format	Meaning
`lb rd, imm[11:0](rs1)`	I	Load byte, signed
`lbu rd, imm[11:0](rs1)`	I	Load byte, unsigned
`lh rd, imm[11:0](rs1)`	I	Load half-word, signed
`lhu rd, imm[11:0](rs1)`	I	Load half-word, unsigned
`lw rd, imm[11:0](rs1)`	I	Load word
`sb rs2, imm[11:0](rs1)`	S	Store byte
`sh rs2, imm[11:0](rs1)`	S	Store half-word
`sw rs2, imm[11:0](rs1)`	S	Store word
`fence pred, succ`	I	Memory ordering fence
`fence.i`	I	Instruction memory ordering fence

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

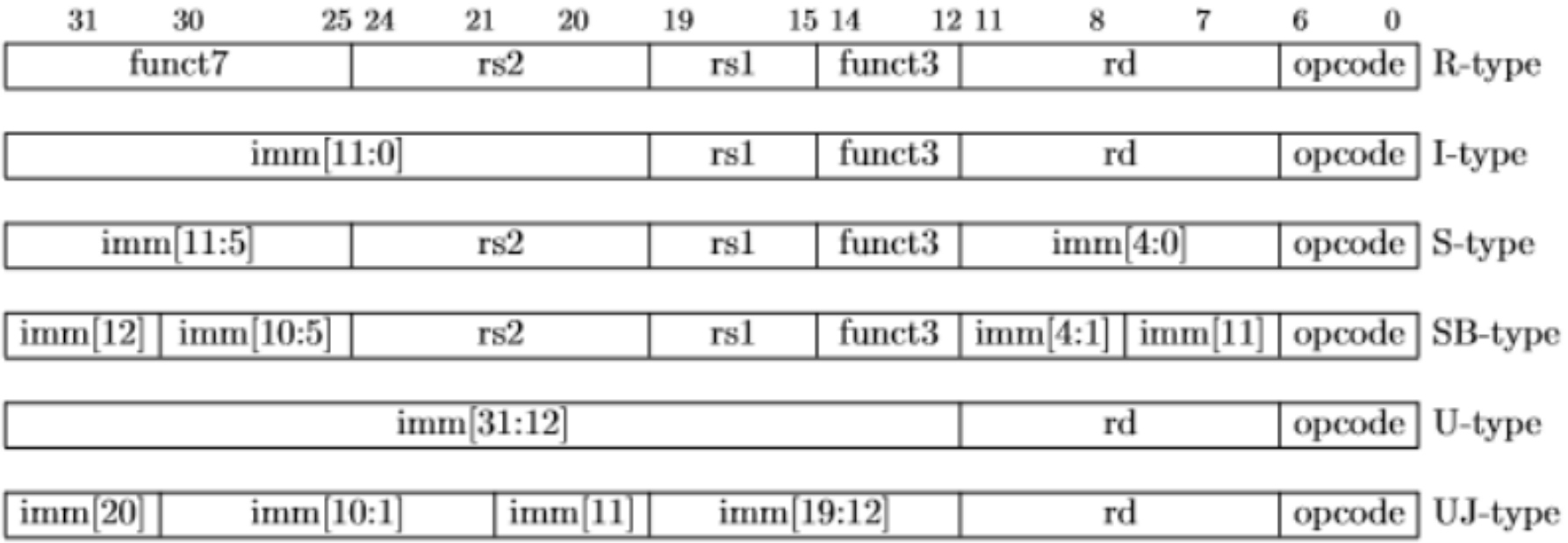
RV32I: Instruções (Controle)

Instruction	Format	Meaning
`beq rs1, rs2, imm[12:1]`	SB	Branch if equal
`bne rs1, rs2, imm[12:1]`	SB	Branch if not equal
`blt rs1, rs2, imm[12:1]`	SB	Branch if less than, 2's complement
`bltu rs1, rs2, imm[12:1]`	SB	Branch if less than, unsigned
`bge rs1, rs2, imm[12:1]`	SB	Branch if greater or equal, 2's complement
`bgeu rs1, rs2, imm[12:1]`	SB	Branch if greater or equal, unsigned
`jal rd, imm[20:1]`	UJ	Jump and link
`jalr rd, rs1, imm[11:0]`	I	Jump and link register

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Formatos das Instruções



Arquitetura RISC-V

RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
0000 | pred	succ	00000	000	00000	0001111	FENCE
0000 | 0000	0000	00000	001	00000	0001111	FENCE.I
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Registradores

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5–7	t0–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller

RISC-V Pseudo instruções

Pseudoinstruction	Base Instruction(s)	Meaning
`la rd, symbol`	`auipc rd, symbol[31:12]` `addi rd, rd, symbol[11:0]`	Load address
`l{b|h|w|d} rd, symbol`	`auipc rd, symbol[31:12]` `l{b|h|w|d} rd, symbol[11:0](rd)`	Load global
`s{b|h|w|d} rd, symbol, rt`	`auipc rt, symbol[31:12]` `s{b|h|w|d} rd, symbol[11:0](rt)`	Store global
`fl{w|d} rd, symbol, rt`	`auipc rt, symbol[31:12]` `fl{w|d} rd, symbol[11:0](rt)`	Floating-point load global
`fs{w|d} rd, symbol, rt`	`auipc rt, symbol[31:12]` `fs{w|d} rd, symbol[11:0](rt)`	Floating-point store global
`nop`	`addi x0, x0, 0`	No operation
`li rd, immediate`	*Myriad sequences*	Load immediate
`mv rd, rs`	`addi rd, rs, 0`	Copy register
`not rd, rs`	`xori rd, rs, -1`	One's complement
`neg rd, rs`	`sub rd, x0, rs`	Two's complement
`negw rd, rs`	`subw rd, x0, rs`	Two's complement word
`sext.w rd, rs`	`addiw rd, rs, x0`	Sign extend word
`seqz rd, rs`	`sltiu rd, rs, 1`	Set if = zero
`snez rd, rs`	`sltu rd, x0, rs`	Set if ≠ zero
`sltz rd, rs`	`slt rd, rs, x0`	Set if < zero
`sgtz rd, rs`	`slt rd, x0, rs`	Set if > zero
`fmv.s rd, rs`	`fsgnj.s rd, rs, rs`	Copy single-precision register
`fabs.s rd, rs`	`fsgnjx.s rd, rs, rs`	Single-precision absolute value
`fneg.s rd, rs`	`fsgnjn.s rd, rs, rs`	Single-precision negate
`fmv.d rd, rs`	`fsgnj.d rd, rs, rs`	Copy double-precision register
`fabs.d rd, rs`	`fsgnjx.d rd, rs, rs`	Double-precision absolute value
`fneg.d rd, rs`	`fsgnjn.d rd, rs, rs`	Double-precision negate
`beqz rs, offset`	`beq rs, x0, offset`	Branch if ≠ zero
`bnez rs, offset`	`bne rs, x0, offset`	Branch if = zero
`blez rs, offset`	`bge x0, rs, offset`	Branch if ≤ zero
`bgez rs, offset`	`bge rs, x0, offset`	Branch if ≥ zero
`bltz rs, offset`	`blt rs, x0, offset`	Branch if < zero
`bgtz rs, offset`	`blt x0, rs, offset`	Branch if > zero
`j offset`	`jal x0, offset`	Jump
`jal offset`	`jal x1, offset`	Jump and link
`jr rs`	`jalr x0, rs, 0`	Jump register
`jalr rs`	`jalr x1, rs, 0`	Jump and link register
`ret`	`jalr x0, x1, 0`	Return from subroutine
`call offset`	`auipc x6, offset[31:12]` `jalr x1, x6, offset[11:0]`	Call far-away subroutine
`tail offset`	`auipc x6, offset[31:12]` `jalr x0, x6, offset[11:0]`	Tail call far-away subroutine

Organização e Arquitetura Básicas de Computadores

RV32I: Exemplos de Execução de Instruções (Assembly)

```
addi a0, zero, 12      # a0 <- 12
addi a1, zero, 4       # a1 <- 4
 lui a0, (0x12345000>>12)
 ori a0, a0, 0x678       # a0 <- 0x12345678
addi a1, zero, 255     # a1 <- 255
```

Organização e Arquitetura Básicas de Computadores

RV32I: Exemplos de Execução de Instruções (Assembly)

```
00011db8 <__register_exitproc>:
   11db8:       fe010113                addi    sp,sp,-32
   11dbc:       00812c23                sw      s0,24(sp)
   11dc0:       8041a403                lw      s0,-2044(gp) # 1c454 <_global_impure_ptr>
   11dc4:       00912a23                sw      s1,20(sp)
   11dc8:       00112e23                sw      ra,28(sp)
   11dcc:       14842783                lw      a5,328(s0)
   11dd0:       00050493                mv      s1,a0
   11dd4:       0c078863                beqz    a5,11ea4 <__register_exitproc+0xec>
   11dd8:       0047a703                lw      a4,4(a5)
   11ddc:       01f00513                li      a0,31
   11de0:       02e54a63                blt     a0,a4,11e14 <__register_exitproc+0x5c>
   11de4:       00170513                addi    a0,a4,1
   11de8:       00271813                slli    a6,a4,0x2
   11dec:       08049063                bnez    s1,11e6c <__register_exitproc+0xb4>
   11df0:       01078833                add     a6,a5,a6
   11df4:       00a7a223                sw      a0,4(a5)
   11df8:       00000513                li      a0,0
   11dfc:       00b82423                sw      a1,8(a6)
   11e00:       01c12083                lw      ra,28(sp)
   11e04:       01812403                lw      s0,24(sp)
   11e08:       01412483                lw      s1,20(sp)
   11e0c:       02010113                addi    sp,sp,32
   11e10:       00008067                ret
```

Organização e Arquitetura Básicas de Computadores

RV32I: Exemplos de Execução de Instruções (Assembly)

```
int a[64];
for (int i = 0; i < 64; i++)
  a[i] += 1;
```

```
        move    x3,x0
        li      x6,64
$LOOP:  lw      x5,0(x4)
        addw    x2,x3,1
        move    x3,x2
        addw    x5,x5,1
        sw      x5,0(x4)
        add     x4,x4,4
        bne     x2,x6,$LOOP
```

Organização e Arquitetura Básicas de Computadores

Mais Informações:

https://riscv.org/

RISC-V (Simuladores e outras ferramentas)

Página de LOAC/DSC/CEEI/UFCG