Arquitetura e Organização de Computadores
Turma C 2020/02

Trabalho: Geração de Dados Imediatos no RISC-V

Objetivo:

Desenvolver um módulo em VHDL que gere os dados imediatos utilizados nas instruções do processador RISC-V.

Descrição:

O conjunto de instruções do processador RISC-V introduz várias alternativas para geração de dados imediatos, ou seja, dados que são incluídos no próprio código da instrução.

Os formatos de instrução utilizados são R, I, S, SB, U e UJ. Os campos utilizados nestes formatos são indicados a seguir.

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R-type	funct7							rs2					rs1					funct3			rd					opcode						
I-type	11	10	9	8	7	6	5	4	3	2	1	0	rs1					funct3			rd					opcode						
S-type	11	10	9	8	7	6	5	rs2					rs1					funct3			4	3	2	1	0	opcode						
SB-type	12	10	9	8	7	6	5	rs2					rs1					funct3			4	3	2	1	11	opcode						
UJ-type	20	10	9	8	7	6	5	4	3	2	1	11	19	18	17	16	15	14	13	12	rd					opcode						
U-type	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	rd					opcode						

Figura 1. Formatos de instruções do RISC-V

Campos principais:

funct7: campo de 7 bits auxiliar na identificação da instrução
rs2: registrador fonte da operação
rs1: registrador fonte da operação
rd: registrados destino da operação
funct3: campo de 3 bits auxiliar na identificação da instrução
opcode: campo de 7 bits que identifica a instrução

Dados Imediatos (considerando ins = instrução, imm = dado imediato gerado):

- I-type: 12 bits, *imm(11:0) = ins(31:20)* e *imm(31:12)* = extensão de sinal.
- S-type: 12 bits, *imm(11:0)* = {*ins(31:25), ins(11:7)*} e *imm(31:12)* = extensão de sinal.
- SB-type: 12 bits, *imm(12:1)* = {*ins(31), ins(7), ins(30:25), ins(11:8)*}, sendo *imm(0) = 0* e *imm(31:13)* = extensão de sinal.
- UJ-type: 20 bits, *imm(20:1)* = {*ins(31), ins(19:12), ins(20), ins(30:21)*}, sendo *imm(0)* = 0 e *imm(31:20)* = extensão de sinal.
- U-type: 20 bits, *imm(31:12),* = *ins(31:12)* e os *imm(11:0)* = 0.

Identificação dos formatos:

```
type FORMAT_RV is { R_type, I_type, S_type,  SB_type, UJ_type, U_type };
```

Os formatos são definidos pelos seguintes opcodes:

- R_type: opcode = 0x33
- I_type: opcode = 0x03 ou opcode = 0x13 ou opcode = 0x67
- S_type: opcode = 0x23
- SB_type: opcode = 0x63
- U_type: opcode = 0x37
- UJ_type: opcode = 0x6F

Tarefa:

Desenvolver em *VHDL* um módulo que recebe como entrada uma instrução de 32 bits em um dos formatos acima indicados e gera o valor do dado imediato, também de 32 bits, em sua saída. No caso do formato *R-type*, que não inclui dado imediato na instrução, gerar o valor zero.

Entradas e saídas:

```
entity genImm32 is
        port (
              instr :       in std_logic_vector(31 downto 0);
              imm32 :       out signed(31 downto 0));
end genImm32;
```

Verificação:

Para verificar o funcionamento do módulo, utilizar as seguintes instruções do RISC-V:

Instrução RISC-V	Código	Formato	Imediato
add t0, zero, zero	0x000002b3	R-type	inexiste: 0
lw t0, 16(zero)	0x01002283	I-type0	16
addi t1, zero, -100	0xf9c00313	I-type1	-100
xori t0, t0, -1	0xfff2c293	I-type1	-1
addi t1, zero, 354	0x16200313	I-type1	354
jalr zero, zero, 0x18	0x01800067	I-type2	0x18 / 24
lui s0, 2	0x00002437	U-type	0x2000
sw t0, 60(s0)	0x02542e23	S-type	60
bne t0, t0, main	0xfe5290e3	SB-type	-32_c
jal rot	0x00c000ef	UJ-type	0xC / 12

Comandos VHDL:

- agregação: operador ‘&’
- ex:

```
architecture a of genImm32 is
        signal a, b, d : std_logic_vector(31 downto 0);
begin
        a <= b(15 downto 1) & d(31 downto 16) & ‘0’;
        …
end gemImm32;
```