ThinPad 设计概要

涂珂 2011011273
傅左右 2011011264
计 14 - 402 组

November 21, 2013

Contents

概述

本次实验设计支持指令流水的 CPU，并设计了转发单元（旁路回路）解决数据冲突，以及冒险检测单元解决控制冲突。

指令集任务

THCO MIPS 基本指令集

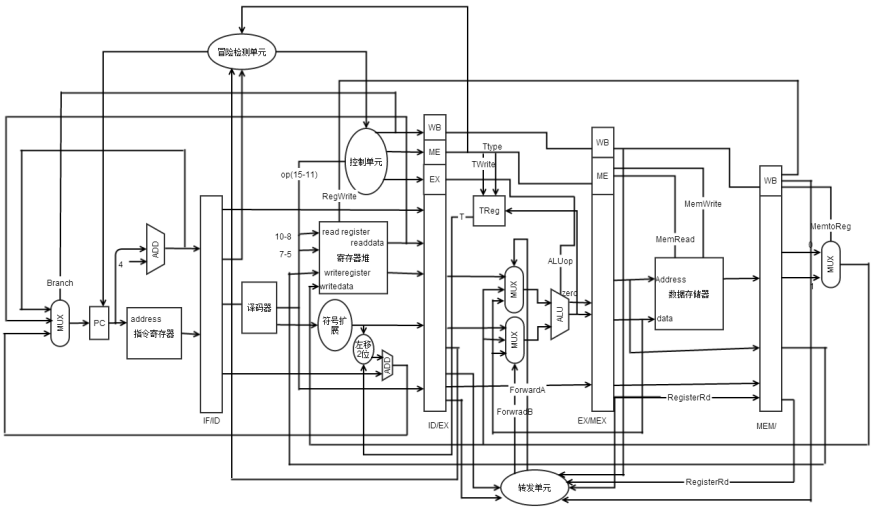
序号	指令	序号	指令
1	ADDIU	14	LW_SP
2	ADDIU3	15	MFIH
3	ADDSP	16	MFPC
4	ADDU	17	MTIH
5	AND	18	MTSP
6	B	19	NOP
7	BEQZ	20	OR
8	BNEZ	21	SLL
9	BTEQZ	22	SRA
10	CMP	23	SUBU
11	JR	24	SW
12	LI	25	SW_SP
13	LW		

扩展指令集 (402)

- JRRA
- SLTI
- ADDSP3
- NOT
- SLT

数据通路

数据通路和调用关系可见datapath.png文件。



数据通路示意图

指令设计

- 用前 5 位表示 op。共 30 条。
- 加 * 为扩展指令。
- XXX，YYY，ZZZ 为寄存器标号。
- III 为立即数。
- 把类型相近的 op 连续起来，这样写代码就可以用大于小于判断了。

R 型指令

R	指令结构
MFIH	00001XXX00000000
MFPC	00010XXX00000000
MTIH	00011XXX00000000
MTSP	00100XXX00000000
AND	00101XXXYYY00000

OR	00110XXXYYY00000
*NOT	00111XXXYYY00000
*SLT	01000XXXYYY00000
CMP	01001XXXYYY00000
SLL	01010XXXYYYIII00
SRA	01011XXXYYYIII00
ADDU	01100XXXYYYZZZ00
SUBU	01101XXXYYYZZZ00

I 型指令

I	指令结构
ADDSP	01110IIIIIIII000
LW_SP	01111XXX00000000
ADDIU	10000XXXIIIIIIII
*SLTI	10001XXXIIIIIIII
*ADDSP3	10010XXXIIIIIIII
LI	10011XXXIIIIIIII
ADDIU3	10100XXXYYY0IIII
LW	10101XXXYYYIIIII
SW	10110XXXYYYIIIII
SW_SP	10111XXXYYYIIIII

B 型指令

B	指令结构
B	11000IIIIIIIIII
BTEQZ	11001IIIIIIII000
BEQZ	11010XXXIIIIIIII
BNEZ	11011XXXIIIIIIII

J 型指令

J	指令结构
*JRRA	1110000000000000
JR	11101XXX00000000

NOP 指令

NOP	0000000000000000

指令执行

唔，由于我们把指令格式、指令的执行流程和指令关联信号都放在指令设计表格中，表格有点长，文档中放不下，还是烦请参阅`instruction.xlsx`文件。

每一条指令在`instruction.xlsx`有如下描述：

- 指令类型
  R、I、B、J、NOP
- 指令编码
  我们重新设计了指令的编码（就是前面列举的表格）和汇编器。
- 指令格式（描述性）
- 指令功能（描述性）
- 指令在每一个阶段具体的执行流程
  - IF
  - ID
  - EX
  - MEM
  - WB
- 与指令相关的控制信号
  - ALUop
  - ALUsrc
  - Ttype

- Twrite
- MemRead
- MemWrite
- MemtoReg

控制信号设计

(可参阅 xlsx 文件`signal.xlsx`。)

控制信号	发生阶段	置 0 时	置 1 时	置 10 时
PCWrite	IF	null	写 PC	
Branch(2 位)	ID	PC+4	PC+4+immediate	Reg1
ForwardA(2 位)	EX	来自寄存器堆的输出 Reg1	转发写回的值	转发上一次 ALU 运算结果
ForwardB(2 位)	EX	来自寄存器堆的输出 Reg2	转发写回的值	转发上一次 ALU 运算结果
ALUsrc	EX	来自寄存器堆的输出	来自符号扩展的立即数	无
ALUop(3 位)	EX	加	001: 减,010: 与, 011: 或，100: 非, 101: 左移,110: 右移	无
Ttype	EX	小于 T 为 1	不等于 T 为 1	
Twrite	EX	null	写入 T	
MemRead	MEM	null	读内存	
MemWrite	MEM	null	写内存	
MemtoReg	WB	写入 ALU 输出值	写入内存输入值	

模块接口设计

寄存器堆 - RegFile

Listing 1: 寄存器堆 - RegFile

```
entity RegFile is
    Port ( ReadAddress1 : in  STD_LOGIC_VECTOR (3 downto 0);
           ReadAddress2 : in  STD_LOGIC_VECTOR (3 downto 0);
           WriteAddress : in  STD_LOGIC_VECTOR (3 downto 0);
           WriteData : in  STD_LOGIC_VECTOR (15 downto 0);
           Reg1 : out  STD_LOGIC_VECTOR (15 downto 0);
           Reg2 : out  STD_LOGIC_VECTOR (15 downto 0);
           RegWrite : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           rst : in  STD_LOGIC);
end RegFile;
```

运算

ALU

Listing 2: ALU

```
entity ALU is
    Port ( Input1 : in  STD_LOGIC_VECTOR (15 downto 0);
           Input2 : in  STD_LOGIC_VECTOR (15 downto 0);
           Output : out  STD_LOGIC_VECTOR (15 downto 0);
           ALUop : in  STD_LOGIC_VECTOR (2 downto 0));
end ALU;
```

Add

Listing 3: Add

```
entity Add is
    Port ( Input1 : in  STD_LOGIC_VECTOR (15 downto 0);
           Input2 : in  STD_LOGIC_VECTOR (15 downto 0);
           Output : out  STD_LOGIC_VECTOR (15 downto 0));
end Add;
```

Mux

Listing 4: Mux

```
entity Mux is
    Port ( choice : in  STD_LOGIC_VECTOR (1 downto 0);
           Input1 : in  STD_LOGIC_VECTOR (15 downto 0);
           Input2 : in  STD_LOGIC_VECTOR (15 downto 0);
           Input3 : in  STD_LOGIC_VECTOR (15 downto 0);
           Output : in  STD_LOGIC_VECTOR (15 downto 0));
end Mux;
```

状态寄存器

IF-ID

Listing 5: IF-ID

```
entity IF_ID is
    Port ( Instruction_in : in  STD_LOGIC_VECTOR (15 downto 0);
           Instruction_out : out  STD_LOGIC_VECTOR (15 downto
               0);
           PC_in : in  STD_LOGIC_VECTOR (15 downto 0);
           PC_out : out  STD_LOGIC_VECTOR (15 downto 0);
           clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           WriteIn : in  STD_LOGIC);
end IF_ID;
```

ID_EX

Listing 6: ID-EX

```
entity ID_EX is
    Port ( clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           WriteIn : in  STD_LOGIC;
           ALUopInput : in  STD_LOGIC_VECTOR (2 downto 0);
           ALUsrcInput : in  STD_LOGIC;
           TTypeInput : in  STD_LOGIC;
           TWriteInput : in  STD_LOGIC;
           MemReadInput : in  STD_LOGIC;
           MemWriteInput : in  STD_LOGIC;
           MemtoRegInput : in  STD_LOGIC;
           ALUopOutput : out  STD_LOGIC_VECTOR (2 downto 0);
           ALUsrcOutput : out  STD_LOGIC;
```

```
            TTypeOutput : out  STD_LOGIC;
            TWriteOutput : out  STD_LOGIC;
            MemReadOutput : out  STD_LOGIC;
            MemWriteOutput : out  STD_LOGIC;
            MemtoRegOutput : out  STD_LOGIC;
            DataInput1 : in  STD_LOGIC_VECTOR (15 downto 0);
            DataInput2 : in  STD_LOGIC_VECTOR (15 downto 0);
            ImmediateInput : in  STD_LOGIC_VECTOR (15 downto 0);
            ALUdata1 : out  STD_LOGIC_VECTOR (15 downto 0);
            ALUdata2 : out  STD_LOGIC_VECTOR (15 downto 0);
            ImmediateOutput : out  STD_LOGIC_VECTOR (15 downto
                0);
            RegReadInput1 : in  STD_LOGIC_VECTOR (3 downto 0);
            RegReadInput2 : in  STD_LOGIC_VECTOR (3 downto 0);
            RegWriteInput : in  STD_LOGIC_VECTOR (3 downto 0);
            RegReadOutput1 : out  STD_LOGIC_VECTOR (3 downto 0);
            RegReadOutput2 : out  STD_LOGIC_VECTOR (3 downto 0);
            RegWriteOutput : out  STD_LOGIC_VECTOR (3 downto 0))
                ;
end ID_EX;
```

EX-MEM

Listing 7: EX-MEM

```
entity EX_MEM is
    Port ( clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           WriteIn : in  STD_LOGIC;
           MemReadInput : in  STD_LOGIC;
           MemWriteInput : in  STD_LOGIC;
           MemtoRegInput : in  STD_LOGIC;
           MemReadOutput : out  STD_LOGIC;
           MemWriteOutput : out  STD_LOGIC;
           MemtoRegOutput : out  STD_LOGIC;
           DataInput : in  STD_LOGIC_VECTOR (15 downto 0);
           DataOutput : out  STD_LOGIC_VECTOR (15 downto 0);
              RegReadInput1 : in  STD_LOGIC_VECTOR (3 downto 0)
                  ;
           RegReadInput2 : in  STD_LOGIC_VECTOR (3 downto 0);
           RegWriteInput : in  STD_LOGIC_VECTOR (3 downto 0);
           RegReadOutput1 : out  STD_LOGIC_VECTOR (3 downto 0);
           RegReadOutput2 : out  STD_LOGIC_VECTOR (3 downto 0);
           RegWriteOutput : out  STD_LOGIC_VECTOR (3 downto 0))
               ;
end EX_MEM;
```

MEM-WB

Listing 8: MEM-WB

```
entity MEM_WB is
    Port ( clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           WriteIn : in  STD_LOGIC;
           MemtoRegInput : in  STD_LOGIC;
           MemtoRegOutput : out  STD_LOGIC;
           AluResultInput : in  STD_LOGIC_VECTOR (15 downto 0);
           AluResultOutput : out  STD_LOGIC_VECTOR (15 downto
               0);
              RegReadInput1 : in  STD_LOGIC_VECTOR (3 downto 0)
                  ;
           RegReadInput2 : in  STD_LOGIC_VECTOR (3 downto 0);
           RegWriteInput : in  STD_LOGIC_VECTOR (3 downto 0);
           RegReadOutput1 : out  STD_LOGIC_VECTOR (3 downto 0);
           RegReadOutput2 : out  STD_LOGIC_VECTOR (3 downto 0);
           RegWriteOutput : out  STD_LOGIC_VECTOR (3 downto 0))
               ;
end MEM_WB;
```

寄存器

PC 寄存器 - PCReg

Listing 9: PC 寄存器 - PCReg

```
entity PCReg is
    Port ( Input : in  STD_LOGIC_VECTOR (15 downto 0);
           Output : out  STD_LOGIC_VECTOR (15 downto 0);
           clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           PCWrite : in  STD_LOGIC);
end PCReg;
```

T 寄存器 - TReg

Listing 10: T 寄存器 - TReg

```
entity TReg is
    Port ( Input : in  STD_LOGIC_VECTOR (15 downto 0);
           TType : in  STD_LOGIC;
           TWrite : in  STD_LOGIC;
```

```
           T : out  STD_LOGIC);
end TReg;
```

存储器

数据存储器 - DataMem

Listing 11: 数据存储器 - DataMem

```
entity DataMem is
    Port ( Address : in  STD_LOGIC_VECTOR (15 downto 0);
           Input : in  STD_LOGIC_VECTOR (15 downto 0);
           Output : out  STD_LOGIC_VECTOR (15 downto 0);
           MemWrite : in  STD_LOGIC;
           MemRead : in  STD_LOGIC);
end DataMem;
```

指令存储器 - InstructionMem

Listing 12: 指令存储器 - InstructionMem

```
entity InstructionMem is
    Port ( Address : in  STD_LOGIC_VECTOR (15 downto 0);
           Data : out  STD_LOGIC_VECTOR (15 downto 0));
end InstructionMem;
```

译码器

Listing 13: 译码器 - Decoder

```
entity Decoder is
    Port ( Instruction : in  STD_LOGIC_VECTOR (15 downto 0);
           Op : out  STD_LOGIC_VECTOR (4 downto 0);
           Reg1 : out  STD_LOGIC_VECTOR (3 downto 0);
           Reg2 : out  STD_LOGIC_VECTOR (3 downto 0);
           Reg3 : out  STD_LOGIC_VECTOR (3 downto 0);
           Imm : out  STD_LOGIC_VECTOR (15 downto 0));
end Decoder;
```

控制器 Controller

Listing 14: 控制器 - Controller

```
entity Controller is
    Port ( Op : in  STD_LOGIC_VECTOR (4 downto 0);
           rst : in  STD_LOGIC;
           Branch : out  STD_LOGIC_VECTOR (1 downto 0);
           ALUop : out  STD_LOGIC_VECTOR (2 downto 0);
           TType : out  STD_LOGIC;
           TWrite : out  STD_LOGIC;
           MemRead : out  STD_LOGIC;
           MemWrite : out  STD_LOGIC;
           MemtoReg : out  STD_LOGIC);
end Controller;
```

冲突检查

数据冲突的转发单元 Passer（其实应该是 forwarding 吧 =。=）

Listing 15: 数据冲突的转发单元 Passer

```
entity Passer is
    Port ( EXMEM_RegWrite : in  STD_LOGIC;
           MEMWB_RegWrite : in  STD_LOGIC;
           EXMEM_W : in  STD_LOGIC_VECTOR (3 downto 0);
           MEMWB_W : in  STD_LOGIC_VECTOR (3 downto 0);
           IDEX_R1 : in  STD_LOGIC_VECTOR (3 downto 0);
           IDEX_R2 : in  STD_LOGIC_VECTOR (3 downto 0);
           ForwardA : out  STD_LOGIC_VECTOR (2 downto 0);
           ForwardB : out  STD_LOGIC_VECTOR (2 downto 0));
end Passer;
```

控制冲突 RiskChecker

Listing 16: 控制冲突 RiskChecker

```
entity RiskChecker is
    Port ( PCWrite : out  STD_LOGIC;
           IFIDWrite : out  STD_LOGIC;
           ControlRst : out  STD_LOGIC;
           IDEX_MemWrite : in  STD_LOGIC;
           IDEX_W : in  STD_LOGIC;
           IDEX_R1 : in  STD_LOGIC;
           IDEX_R2 : in  STD_LOGIC);
end RiskChecker;
```