计算机组成原理 - 大实验
实验报告

涂珂 2011011273 计 14　　傅左右 2011011264 计 14

December 12, 2013

Contents

1 实验目标

基于 THINPAD 教学计算机，设计：

- 基于 MIPS16 指令集的流水线 CPU
- 使用基本存储、扩展存储、Flash、IO 设备
- 能够运行 kernel、监控程序、project1 程序

2 指令集任务

2.1 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		

2.2 扩展指令集 (402)

- JRRA
- SLTI
- ADDSP3
- NOT
- SLT

3 实验成果指标

- CPU 主频为 6.25MHz（12.5MHz 有时会出一些问题，所以只能二分之，6.25MHz 是稳定频率）
- VGA 分辨率为 640*480
- **继续加啊**

4 实验成果简列

- 清晰的模块分工
- 指令集改进，指令集汇编工具
- 数据旁路
- 冒险检测
- 完整 VGA 调试工具
- FLASH 自启动
- 地址映射统一管理 IO 设备
- 串口通信
- VGA、键盘交互
  - VGA 等宽 ASCII 字符集显示
  - VGA 双端 FIFO 显存
  - 键盘输入、支持换行、发送串口与 VGA 的记事本程序

5 整体设计图

加图

6 重新设计的指令集

6.1 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。

6.2 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

6.3 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

6.4 B 型指令

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

6.5 J 型指令

J	指令结构
*JRRA	1110000000000000
JR	11101XXX00000000

6.6 NOP 指令

NOP	0000000000000000

6.7 指令流水细节

关于每一条指令在流水的五个步骤中具体做了什么。表格无法正常显示下。请见相关设计文档 `instruction.xlsx`。

7 统一的信号及编码

7.1 控制信号

每一级流水阶段的寄存器均会储存相应信号。(显然的，当前指令与当前流水信号相对应)

Table 7: 控制信号表

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	
ALUop(3 位)	EX	加	001: 减, 010: 与, 011: 或, 100: 非, 101: 左移,110: 右移	
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	
RegWrite	WB	null	写入寄存器堆	

7.2 寄存器编址

我们将特殊寄存器也看作普通的寄存器。将寄存器编码长度（3 位）扩长一位，与特殊寄存器统一编址（4 位）。这样在 CPU 内部处理的时候就能简化流程。

我们设置了 SP、PC、RA、IH、Zero 四个特殊寄存器。8 个普通寄存器。特殊寄存器高位均为 1。

Table 8: 寄存器编址表

编码 4 位	寄存器
0+XXX	8 个普通寄存器
1000	Zero
1010	PC
1011	IH
1100	RA
1101	SP

7.3 字符编码

我们将 95 个 ASCII 可打印字符的等宽字符写进了 VGA 控制模块的存储里。我们以空格（0x20）为偏移量，将每一个字符对应的 ASCII 码减去 0x20 作为内部字符编码。这样就有两个好处：一是减少地址量，二是可以用连接两个 std_logic_vector 的方法（`std_logic_vector & std_logic_vector`）计算字符地址。

7.4 指令集与控制信号关系表

Table 9: 指令与控制信号关系

指令	Branch	ALU-op	ALU-src	T-type	T-write	Mem-Read	Mem-Write	Mem-to-Reg	Reg-Write
NOP	00	000	0	0	0	0	0	0	0
MFIH	00	000	0	0	0	0	0	0	1
MFPC	00	000	0	0	0	0	0	0	1
MTIH	00	000	0	0	0	0	0	0	1
MTSP	00	000	0	0	0	0	0	0	1
AND	00	010	0	0	0	0	0	0	1
OR	00	011	0	0	0	0	0	0	1
NOT*	00	100	0	0	0	0	0	0	1
SLT*	00	001	0	0	1	0	0	0	0
CMP	00	001	0	1	1	0	0	0	0
SLL	00	101	1	0	0	0	0	0	1
SRA	00	110	1	0	0	0	0	0	1
ADDU	00	000	0	0	0	0	0	0	1
SUBU	00	001	0	0	0	0	0	0	1
ADDSP	00	000	1	0	0	0	0	0	1
LW_SP	00	000	1	0	0	1	0	1	1
SW_SP	00	000	1	0	0	0	1	0	0
ADDIU	00	000	1	0	0	0	0	0	1
SLTI*	00	001	1	0	1	0	0	0	0
ADDSP3*	00	000	1	0	0	0	0	0	1
LI	00	000	1	0	0	0	0	0	1
ADDIU3	00	000	1	0	0	0	0	0	1
LW	00	000	1	0	0	1	0	1	1
SW	00	000	1	0	0	0	1	0	0
B	01	000	0	0	0	0	0	0	0
BTEQZ	01	000	0	0	0	0	0	0	0
BEQZ	01	000	0	0	0	0	0	0	0
BNEZ	01	000	0	0	0	0	0	0	0
JRRA*	10	000	0	0	0	0	0	0	0
JR	10	000	0	0	0	0	0	0	0