单周期CPU设计

李厚润2019202463

单周期数据通路设计以及合并

指令	Adder	Adder	PC	IM	REG	REG	REG	REG	ALU	ALU	DM	DM	Nadd	Nadd	shift
指令	A	B	PC	addr	reg1	reg2	Wreg	Wdata	A	B	Addr	Wdata	A	B	imm
R型指令	PC	4	Adder	PC	Rs	Rt	Rd	ALU	Rdata1	Rdata2					
addi	PC	4	Adder	PC	Rs		Rt	ALU	Rdata1	signedExt16					
LW	PC	4	Adder	PC	Rs		Rt	DM	Rdata1	signedExt16	ALU				
SW	PC	4	Adder	PC	Rs	Rt			Rdata1	signedExt16	ALU	Rdata2			
Beq	PC	4	Adder|Nadd	PC	Rs	Rt			Rdata1	Rdata2			Adder	shift	imm16
lui	PC	4	Adder	PC			Rt	ALU		imm16					
ori	PC	4	Adder	PC	Rs		Rt	ALU	Rdata1	imm16					
j	PC	4	shift26	PC											imm26
jal	PC	4	shift26				GR[31]	Nadd					Adder	4	imm26
jr	PC	4	Rdata1	PC	Rs										
syscall															
合并	PC	4	Adder|Nadd|shift|Rdata1	PC	Rs	Rt	Rd|Rt|GR[31]	ALU|DM|Nadd	Rdata1	Rdata2|imm16|signedExt16	ALU	Rdata2	Adder	shift|4	imm

主控单元控制信号分析

我们用：

- RegDst 表示向GeneralPurposeRegisters 写入的目的寄存器
- ALUSrc 表示ALU第二个操作数B的来源
- MemtoReg 表示向GeneralPurposeRegisters 写入的数据
- Branch表示跳转指令的类型
- ALUOP表示ALU操作码

1. RegDst wire[1:0] for wreg

- jal指令：RegDst = 2， 选择GR[31]
- R型指令：RegDst = 1，选择Rd
- 其他：RegDst = 0，选择Rt

2. ALUSrc wire[1:0] for ALUB

- R型指令或者Beq：ALUSrc = 0， 选择寄存器的Rdata2作为输入
- lui，ori：ALUSrc = 1， 选择立即数imm16作为输入
- 其他：ALUSrc = 2， 选择立即数signedExt16作为输入

3. MemtoReg wire[1:0] for wdata

- jal指令：MemtoReg = 2， 选择Nadd输出 (pc+8)
- LW指令：MemtoReg = 1，选择数据存储器DM输出
- 其他：MemtoReg = 0，选择ALU输出

4. Branch wire[1:0]

- jal，j指令：Branch=`2'b11`, PC设置为shift26
- jr指令：Branch=`2'b10`, PC设置为Rdata1
- Beq指令：Branch=`2’b01`，此时若Zero=1， PC设置为选择加法器Nadd 输出(分支指令目的地址)，否则设置为Add输出(PC+4)
- 其他指令:Branch=`2’b00`， PC设置为加法器Add输出(PC+4)

5. ALUOP wire[2:0]

- ALUOP = `3‘b000` add
- ALUOP = `3‘b001` addu
- ALUOP = `3‘b010` sub
- ALUOP = `3‘b011` subu
- ALUOP = `3‘b100` lui
- ALUOP = `3‘b101` ori

单周期控制器设计

Input/output	signal	R-format(addu)	R-format(subu)	R-format(sub)	addi	LW	SW	Beq	lui	ori	j	jal	jr
Input	OP5	0	0	0	0	1	1	0	0	0	0	0	0
Input	OP4	0	0	0	0	0	0	0	0	0	0	0	0
Input	OP3	0	0	0	1	0	1	0	1	1	0	0	0
Input	OP2	0	0	0	0	0	0	1	1	1	0	0	0
Input	OP1	0	0	0	0	1	1	0	1	0	1	1	0
Input	OP0	0	0	0	0	1	1	0	1	1	0	1	0
input	func5	1	1	1	x	x	x	x	x	x	x	x	0
input	func4	0	0	0	x	x	x	x	x	x	x	x	0
input	func3	0	0	0	x	x	x	x	x	x	x	x	1
input	func2	0	0	0	x	x	x	x	x	x	x	x	0
input	func1	0	1	1	x	x	x	x	x	x	x	x	0
input	func0	1	1	0	x	x	x	x	x	x	x	x	0
output	RegDst	1	1	1	0	0	x	x	0	0	x	2	x
output	ALUSrc	0	0	0	2	2	2	0	1	1	x	x	x
output	MemtoReg	0	0	0	0	1	x	x	0	0	x	2	x
output	Branch1	0	0	0	0	0	0	0	0	0	1	1	1
output	Branch0	0	0	0	0	0	0	1	0	0	1	1	0
output	RegWrite	1	1	1	1	1	0	0	1	1	0	1	0
output	MemRead	0	0	0	0	1	0	0	0	0	0	0	0
output	MemWrite	0	0	0	0	0	1	0	0	0	0	0	0
output	ALUOP2	0	0	0	0	0	0	0	1	1	x	x	x
output	ALUOP1	0	1	1	0	0	0	1	0	0	x	x	x
output	ALUOP0	1	1	0	0	0	0	0	0	1	x	x	x