P5设计文档

思考题

1. 我们使用提前分支判断的方法尽早产生结果来减少因不确定而带来的开销，但实际上这种方法并非总能提高效率，请从流水线冒险的角度思考其原因并给出一个指令序列的例子。

可能beq比较时所需寄存器的值还没有算出来，例如：

```mips
lw $1 0($0)
beq $1 $2 label
```

beq需要等待lw两个周期。

---

2. 因为延迟槽的存在，对于 jal 等需要将指令地址写入寄存器的指令，要写回 PC + 8，请思考为什么这样设计？

由于延迟槽设计，jal的下一条指令一定会执行，所以当返回时应该返回下下一条指令，即PC + 8

---

3. 我们要求大家所有转发数据都来源于流水寄存器而不能是功能部件（如 DM 、 ALU ），请思考为什么？

功能部件的执行时间可能会受多种因素的影响，比如数据的读取速度，指令的复杂程度等。如果直接从功能部件中获取数据，可能会导致数据的延迟，从而影响流水线的性能。因此，我们要求所有转发数据来自流水寄存器。

---

4. 我们为什么要使用 GPR 内部转发？该如何实现？

当D级的指令读取寄存器数据的时候，若该地址的寄存器于W阶段正在写回，若不采用内部转发，读取到的将是旧的数据。采用暂停会影响性能。

```
assign RD1 = (A1 == 0) ? 0 : (A1 == A3) ? WD : register[A1];
assign RD2 = (A2 == 0) ? 0 : (A2 == A3) ? WD : register[A1];
```

---

5. 我们转发时数据的需求者和供给者可能来源于哪些位置？共有哪些转发数据通路？

供给者：E_reg,M_reg,W_reg
需求者：D_NPC,D_CMP,E_ALU,M_DM;

转发通路：

E_reg -> D_NPC,D_CMP
M_reg -> D_NPC,D_CMP,E_ALU
W_reg -> D_NPC,D_CMP,E_ALU,M_DM

---

6. 在课上测试时，我们需要你现场实现新的指令，对于这些新的指令，你可能需要在原有的数据通路上做哪些扩展或修改？提示：你可以对指令进行分类，思考每一类指令可能修改或扩展哪些位置。

主要需要修改的地方应该是CTRL对各个信号的控制，以及新数据的Tuse和Tnew

---

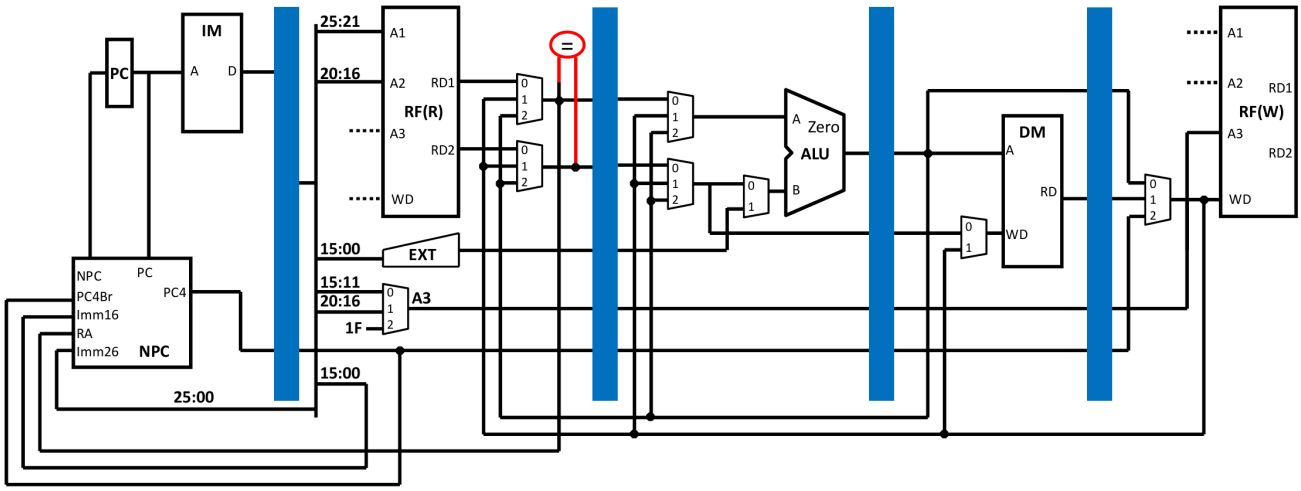
7. 确定你的译码方式，简要描述你的译码器架构，并思考该架构的优势以及不足。

分布式译码。架构是在每一级根据该级的指令单独译码出该级需要的指令，好处流水器寄存器只需要保存该级的指令，不需要保存信号，坏处是要实例化很多译码器

设计方案综述

> F级: IFU D级：CMP,EXT,GRF,NPC,D_reg E级：E_ALU,E_reg M级：M_DM,M_reg W级：W_reg

设计图可以参照：



控制信号

控制信号具体值可以参看译码表。如图：

指令	opcode(31:26)	funct(5:0)	EXTOp	CMPOp	NPCOp	ALUOp	ALUBSel	GRFWDSel	GRFA3Sel	DMWE
功能	opcode	func	立即数扩展	比较器操作	下一条PC计算	ALU运算	ALUB端口选择	GRF写回数据来源	GRF写回地址选择	数据存储器写使能
add	000000	100000	EXT_ZERO	CMP_BEQ	NPC_PC4	ALU_add	ALUBrt	GRFWDALU	GRFA3rd	DMWE_ZERO
sub	000000	100010	EXT_ZERO	CMP_BEQ	NPC_PC4	ALU_sub	ALUBrt	GRFWDALU	GRFA3rd	DMWE_ZERO
ori	001101	x	EXT_ZERO	CMP_BEQ	NPC_PC4	ALU_ori	ALUBimm	GRFWDALU	GRFA3rt	DMWE_ZERO
lw	100011	x	EXT_SIGN	CMP_BEQ	NPC_PC4	ALU_add	ALUBimm	GRFWDDM	GRFA3rt	DMWE_ZERO
sw	101011	x	EXT_SIGN	CMP_BEQ	NPC_PC4	ALU_add	ALUBimm	GRFWDALU	GRFA3rt	DMWE_ONE
beq	000100	x	EXT_ZERO	CMP_BEQ	NPC_BEQ	ALU_sub	ALUBrt	GRFWDALU	GRFA3rt	DMWE_ZERO
lui	001111	x	EXT_ZERO	CMP_BEQ	NPC_PC4	ALU_lui	ALUBimm	GRFWDALU	GRFA3rt	DMWE_ZERO
sll	000000	000000	EXT_ZERO	CMP_BEQ	NPC_PC4	ALU_sll	ALUBrt	GRFWDALU	GRFA3rd	DMWE_ZERO
j	000010	x	EXT_ZERO	CMP_BEQ	NPC_J_Jal	ALU_add	ALUBrt	GRFWDALU	GRFA3rt	DMWE_ZERO

指令	opcode(31:26)	funct(5:0)	EXTOp	CMPOp	NPCOp	ALUOp	ALUBSel	GRFWDSel	GRFA3Sel	DMWE
jal	000011	x	EXT_ZERO	CMP_BEQ	NPC_J_Jal	ALU_add	ALUBrt	GRFWDPC8	GRFA331	DMWE_ZERO
jr	000000	001000	EXT_ZERO	CMP_BEQ	NPC_Jr_Jalr	ALU_add	ALUBrt	GRFWDALU	GRFA3rd	DMWE_ZERO
jalr	000000	001001	EXT_ZERO	CMP_BEQ	NPC_Jr_Jalr	ALU_add	ALUBrt	GRFWDPC8	GRFA3rd	DMWE_ZERO

阻塞与转发

D_Tuse:

指令	add	sub	ori	lw	sw	beq	lui	sll	j	jr	jal	jalr
D_rs_Tuse	1	1	1	1	1	0	1	3	3	0	3	0
D_rt_Tuse	1	1	3	3	2	0	3	1	3	3	3	3

E_Tnew & M_Tnew:

指令	add	sub	ori	lw	sw	beq	lui	sll	j	jr	jal	jalr
E_Tnew	1	1	1	2	0	0	1	1	0	0	0	0
M_Tnew	0	0	0	1	0	0	0	0	0	0	0	0

测试

```
ori $0, $0, 0

ori $1, $1, 1

ori $2, $2, 2

ori $3, $3, 3

ori $4, $4, 4

ori $5, $5, 5

ori $1, $1, 10

ori $2, $2, 20

addu $3, $1, $1

addu $4, $1, $2

subu $5, $2, $1

subu $6, $1, $1

lui $4, 0xffff

nop

beq $1, $0, end

lui $1, 0xffff

end:
```