计算机组成课程设计 P5 实验报告

王肇凯

一、 数据通路

1、F 级组合逻辑

（1）包括 PC、IM、ADD4 模块，以及功能多选器 MUX_PC

（2）PC：将下一条指令的 PC 值赋给当前 PC

IM：读入 code.txt 中的指令，取出下一条指令的操作码

ADD4：将 PC 值+4

MUX_PC：根据 npcsel 信号将下一条指令的 NPC 接入 PC

PC 端口定义：

信号名	方向	描述
PC0[31:0]	I	下一条指令的 PC 值
clk	I	时钟信号
reset	I	复位信号
en	I	使能信号
PC[31:0]	O	当前 PC 值

PC 功能定义：

序号	功能名称	功能描述
1	复位	当时钟上升沿到来时，如果复位信号有效，PC 被复位为 0x00003000
2	输出下一个 PC	当时钟上升沿到来时，将 PC0 赋给 PC

IM 端口定义：

信号名	方向	描述
PC[31:0]	I	当前指令的 PC 值
IR[31:0]	O	当前指令的操作码

IM 功能定义：

序号	功能名称	功能描述
1	读入指令	初始化时将 code.txt 中的指令读入到 im 寄存器
2	输出指令操作码	输出下一条指令的操作码 im[PC[11:2]]

ADD4 端口定义：

信号名	方向	描述
PC[31:0]	I	当前指令的 PC 值
PC4[31:0]	O	PC+4 值

ADD4 功能定义：

序号	功能名称	功能描述
1	计算 ADD4	计算 PC+4 并输出

2、D 级组合逻辑

（1）包括 RF、CMP、EXT、NPC 模块，以及转发多选器 MFCMP1D、MFCMP2D

（2）RF：在 D 级为组合逻辑，根据指令码输出 rs 和 rt 寄存器的值；在 W 级上，时钟上升沿到来且写使能信号有效时，将 WD 写入 A3 寄存器

CMP：beq 指令使用，两输入相等时输出 1，否则输出 0

EXT：根据 EXTop 对 16 位立即数扩展

NPC：根据 npcsel 和 PC 计算出跳转指令的 NPC

MFCMP1D、MFCMP2D：通过转发使 CMP 的输入端、MUX_PC 能获得正确的值

RF 端口定义：

信号名	方向	描述
A1[4:0]	I	rs 寄存器地址
A2[4:0]	I	rt 寄存器地址
A3[4:0]	I	rt/rd 寄存器地址
RegWrite	I	写使能信号
WD[31:0]	I	写入寄存器的数据
clk	I	时钟信号
reset	I	复位信号
RD1[31:0]	O	使能信号
RD2[31:0]	O	当前 PC 值

RF 功能定义：

序号	功能名称	功能描述
1	读取操作数	将 A1、A2 对应寄存器的值输出到 RD1、RD2
2	回写	当时钟上升沿到来时，若写使能信号有效，将 WD 写入 A3 寄存器
3	复位	时钟上升沿到来时，若复位信号有效，则寄存器全部清零

EXT 端口定义：

信号名	方向	描述

imm[15:0]	I	输入的立即数
EXTop[1:0]	I	扩展控制信号
EXTout[31:0]	O	扩展后的立即数

EXT 功能定义：

序号	功能名称	功能描述
1	扩展	EXTop 为`zero_ext 时进行无符号扩展，`sign_ext 时进行有符号扩展，`high_ext 时进行高位扩展

CMP 端口定义：

信号名	方向	描述
D1[31:0]	I	输入数据 1
D2[31:0]	I	输入数据 2
zero	O	比较结果

CMP 功能定义：

序号	功能名称	功能描述
1	比较	D1=D2 时输出 1，否则输出 0

NPC 端口定义：

信号名	方向	描述
PC4[31:0]	I	当前 PC 值
beq	I	npc 控制信号
zero	I	连接 CMP 的输出，判断两数据是否相等
j	I	是否是 j 指令

jal	I	是否是 jal 指令
imm[25:0]	I	26 位操作数
NPC[31:0]	O	下一条指令的 PC 值

NPC 功能定义：

序号	功能名称	功能描述
1	计算下一条指令	当 j 或 jal 为 1 时，输出 PC4[31:28]||imm||00 当 beq 为 1 时，若 zero 为 1，输出 PC4+sign_ext(imm)||00，否则输出 PC4+4 否则输出复位值 0x00003000

3、E 级组合逻辑

（1)包括 ALU 模块以及功能多选器 MUX_ALU、转发多选器 MFALYAE、MFALUBE

（2）MFALUE、MFALUBE 根据指令执行进程选择转发值。MUX_ALU 根据 ALUSrc 选择进入 ALUB 的值

ALU 端口定义：

信号名	方向	描述
A[31:0]	I	操作数 1
B[31:0]	I	操作数 2
ALUctr[1:0]	I	运算控制信号
ALUout[31:0]	O	运算结果

ALU 功能定义：

序号	功能名称	功能描述
1	运算	当 ALUctr 为`add 时，输出两个操作数相加的结果 当 ALUctr 为`sub 时，输出两个操作数相减的结果 当 ALUctr 为`ori 时，输出两个操作数或运算的结果

4、M 级组合逻辑

（1）包括 DM 模块以及 MFDMF 转发多选器

（2）DM 作为内存，MFDMF 转发输出数据

DM 端口定义：

信号名	方向	描述
Addr[31:0]	I	读写内存地址
DIn[31:0]	I	写入内存的数据
MemWrite	I	运算控制信号
clk	I	时钟信号
reset	I	复位信号
PC4	I	当前的 PC+4
DO[31:0]	O	内存中读出的数据

DM 功能定义：

序号	功能名称	功能描述
1	写入内存数据	时钟上升沿且写使能信号有效时，将 WD 写入 dm[Addr[11:2]]中
2	读取内存数据	将 A 代表的地址的数据输出
3	复位	时钟上升沿且复位信号有效时，将 dm 清零

5、W 级组合逻辑

W 级包含一个功能多选器 MUX_WD，用于选择回写到 RF 的数据

6、MUX

（1）功能多选器

名称	控制信号	名字
MUX_PC	npcsel[1:0]	npcsel 为`ADD4 时，PC0=ADD4 npcsel 为`NPC 时，PC0=NPC npcsel 为`MFPCF 时，PC0=MFPCF 否则 PC0=32’h00003000
MUX_ALU	ALUSrc	ALUSrc 为 0 时，将 V2_E 接入 ALUB ALUSrc 为 1 时，将 E32 接入 ALUB
MUX_WD	MemtoReg[1:0]	MemtoReg 为`A0 时，WD=A0_W MemtoReg 为`DR 时，WD=DM_W MemtoReg 为`PC4 时，WD=PC4_W MemtoReg 为`PC8 时，WD=PC4_W+4 否则 WD=0

（2）转发多选器

转发多路选择器名称	控制信号	输入0（低）	输入1	输入2	输入3	输入4（高）	描述
MFCMP1D	MCMP1D[2:0]	DM_W	PC_W	AO_W	PC_M	AO_M	比较单元的第一路输入、jr $rs的跳转地址
MFCMP2D	MCMP2D[2:0]	DM_W	PC_W	AO_W	PC_M	AO_M	比较单元的第二路输入
MFALUBE	MALUBE[2:0]	DM_W	PC_W	AO_W	PC_M	AO_M	ALU 的第二路输入、V2_M
MFALUAE	MALUAE[2:0]	DM_W	PC_W	AO_W	PC_M	AO_M	ALU的第一路输入
MFDMW	MDMW[2:0]	DM_W	PC_W	AO_W			W级DM单元的回写值

二、 控制模块

1、 真值表

	100001	100011								001000	000000
Op	000000	000000	001101	100011	101011	000100	001111	000011	000010	000000	000000
	addu	subu	ori	lw	sw	beq	lui	jal	j	jr	nop
ALUSrc	0	0	1	1	1	0	1	0	0	0	0
MemtoReg[1:0]	`AO	`AO	`AO	`DR	0	0	`AO	`PC8	0	0	0
RegWrite	1	1	1	1	0	0	1	1	0	0	0
MemWrite	0	0	0	0	1	0	0	0	0	0	0
npcsel[1:0]	`ADD4	`ADD4	`ADD4	`ADD4	`ADD4	`NPC	`NPC	`NPC	`NPC	`MFPCF	0
EXTop[1:0]	0	0	`ZERO_EXT	`SIGN_EXT	`SIGN_EXT	0	`high_ext	0	0	0	0

ALUctr[1:0]	`add	`sub	`ori	`add	`add	0	`add	0	0	0	0

2、信号功能

信号名	功能描述
ALUSrc	0：将 RF 的 RD2 输入到 ALU 的 B 接口 1：将 EXT 的 extout 接到 ALU 的 B 接口
MemtoReg[1:0]	`A0：将 ALU 的 ALUout 接到 RF 的 WD 接口  `DR：将 DM 的 dataout 接到 RF 的 WD 接口 `PC8：将 PC8 接到 RF 的 WD 接口
RegWrite	0：不对 RF 进行写操作 1：将 RF 的 WD 写入 A3 中
MemWrite	0:不对 dm 存储器进行写操作 1：将 A0 写入 dm 的 Addr 地址中
npcsel[1:0]	`ADD4:将 ADD4 输入到 PC  `NPC：将 NPC 输入到 PC `MFPCF：将 MFPCF 输入到 PC
EXTop[1:0]	`zero_ext：将立即数进行无符号扩展  `sign_ext：将立即数进行有符号扩展 `high_ext：将立即数进行高位扩展
ALUctr[1:0]	`add：进行加法操作  `sub：进行减法操作 `ori：进行或操作

三、测试程序

```
ori $31,1
addu $30,$31,$31
ori $29,$30,0x6
ori $28,$30,0x8
ori $27,$30,0x10
ori $26,$30,0x12   #addu ori
addu $25,$26,$31
subu $24,$25,$31
subu $23,$25,$24
subu $22,$25,$23
subu $17,$25,$22   #addu subu rs
addu $21,$22,$31
subu $20,$0,$21
subu $19,$20,$21
subu $18,$19,$21

subu $16,$18,$21   #addu subu rt
addu $30,$16,$26
sw $30,0($0)
sw $30,4($0)
sw $30,8($0)
sw $30,12($0)     #addu sw rt
addu $15,$30,$0
beq $15,$30,beq1
nop
beq2:ori $1,1
addu $2,$1,$0
nop
nop
beq $2,$1,beq3
```

```
nop
beq1:ori $1,2
addu $2,$1,$0
nop
beq $2,$1,beq2
nop
beq3:ori $1,3
addu $4,$1,$0
nop
nop

nop
beq $4,$1,beq4
nop
addu $5,$5,$5
beq4:          #addu beq
ori $5,0x30c0
ori $6,4
addu $7,$5,$6
jr $7
nop
addu $5,$5,$5
ori $5,0x30dc
addu $8,$5,$6
nop
nop
jr $8
nop
addu $5,$5,$5
```

```
ori $5,0x30fc
addu $9,$5,$6
nop
nop
nop
jr $9


nop
addu $5,$5,$5
ori $6,$0,5 #addu jr
subu $15,$15,$15
ori $11,4
addu $15,$0,$11
sw $8,0($15)
sw $8,4($15)
sw $8,8($15)    #addu sw rs
ori $4,$0,0
ori $3,4
addu $4,$4,$3
lw $5,0($4)
lw $6,4($4)
lw $7,8($4)       #addu lw rs
```

期望输出：

```
@00003000: $31 <= 00000001
@00003004: $30 <= 00000002
@00003008: $29 <= 00000006
@0000300c: $28 <= 0000000a
@00003010: $27 <= 00000012
@00003014: $26 <= 00000012
@00003018: $25 <= 00000013
@0000301c: $24 <= 00000012
@00003020: $23 <= 00000001
@00003024: $22 <= 00000012
@00003028: $17 <= 00000001
@0000302c: $21 <= 00000013
@00003030: $20 <= ffffffed
@00003034: $19 <= ffffffda
@00003038: $18 <= ffffffc7
@0000303c: $16 <= ffffffb4
@00003040: $30 <= ffffffc6
@00003044: *00000000 <= ffffffc6
@00003048: *00000004 <= ffffffc6
@0000304c: *00000008 <= ffffffc6
@00003050: *0000000c <= ffffffc6
@00003054: $15 <= ffffffc6
@00003078: $ 1 <= 00000002
@0000307c: $ 2 <= 00000002
@00003060: $ 1 <= 00000003
@00003064: $ 2 <= 00000003
@0000308c: $ 1 <= 00000003
@00003090: $ 4 <= 00000003
@000030ac: $ 5 <= 000030c0
@000030b0: $ 6 <= 00000004
@000030b4: $ 7 <= 000030c4
@000030c4: $ 5 <= 000030dc
@000030c8: $ 8 <= 000030e0
@000030e0: $ 5 <= 000030fc
@000030e4: $ 9 <= 00003100
@00003100: $ 6 <= 00000005
@00003104: $15 <= 00000000
@00003108: $11 <= 00000004
@0000310c: $15 <= 00000004
@00003110: *00000004 <= 000030e0
@00003114: *00000008 <= 000030e0
@00003118: *0000000c <= 000030e0
@0000311c: $ 4 <= 00000000
@00003120: $ 3 <= 00000004
@00003124: $ 4 <= 00000004
@00003128: $ 5 <= 000030e0
@0000312c: $ 6 <= 000030e0
@00003130: $ 7 <= 000030e0
```

四、思考题

	Tuse		Tnew	功能部件
	rs	rt		
addu	1	1	1	ALU
subu	1	1	1	ALU
ori	1		1	ALU
lui	1	1	1	ALU
sw	1	2		
lw	1		2	DM
j				
jal			0	PC
jr	0			
beq	0	0		

rs 转发矩阵

	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F

rt 转发矩阵

	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F

前序指令	后续指令	前序指令位置-后续指令位置	处理方式	样例
addu/subu	addu/subu	M-E W-E	转发 转发	addu $5,$1,$2 instr(0,1) subu $6,$5,$3 @
	ori	M-E W-E	转发 转发	addu $5,$1,$2 instr(0,1) ori $6,$5,0xcccc
	beq	E-D M-D W-D	暂停 转发 转发	addu $5,$1,$2 instr(0,1，2) beq $5,$0,loop@
	jr	E-D M-D W-D	暂停 转发 转发	addu $5,$1,$2 instr(0,1，2) jr $5
	sw	W-M	转发	addu $4,$3,$0 sw $4,0($4)

	lw	M-E
		W-E
			转发
			转发
				addu $5,$4,$0
				instr(0,1)
				lw $6,0($5)

ori	addu/subu	M-E W-E	转发 转发	ori $5,$0,0x1111 instr(0,1) addu/subu $6,$5,$5
	ori	M-E W-E	转发 转发	ori $5,$0,0x1111 instr(0,1) ori $6,$5,0xcccc
	beq	E-D M-D W-D	暂停 转发 转发	ori $5,$0,0x1111 instr(0,1,2) beq $5,$1,loop@
	jr	E-D M-D W-D	暂停 转发 转发	ori $5,$0,0x30bc instr(0,1,2) jr $5
	sw	W-M	转发	ori $5,$0,0x1111 sw $5,0($0)
	lw	M-E W-E	转发 转发	ori $5,$4,1 instr(0,1) lw $4,0($5)
lui	addu/subu	M-E W-E	转发 转发	lui $5,0x1111 instr(0,1) addu/subu $6,$5,$5
	ori	M-E W-E	转发 转发	lui $5,0x1111 instr(0,1) ori $6,$5,0xcccc
	beq	E-D M-D W-D	暂停 转发 转发	lui $5, 0x1111 instr(0,1,2) beq $5,$1,loop@
	jr	E-D M-D W-D	暂停 转发 转发	lui $5,0x30bc instr(0,1,2) jr $5
	sw	W-M	转发	lui $5,0x1111 sw $5,0($0)
	lw	M-E W-E	转发 转发	lui $5,0x1111 instr(0,1) lw $40($5)

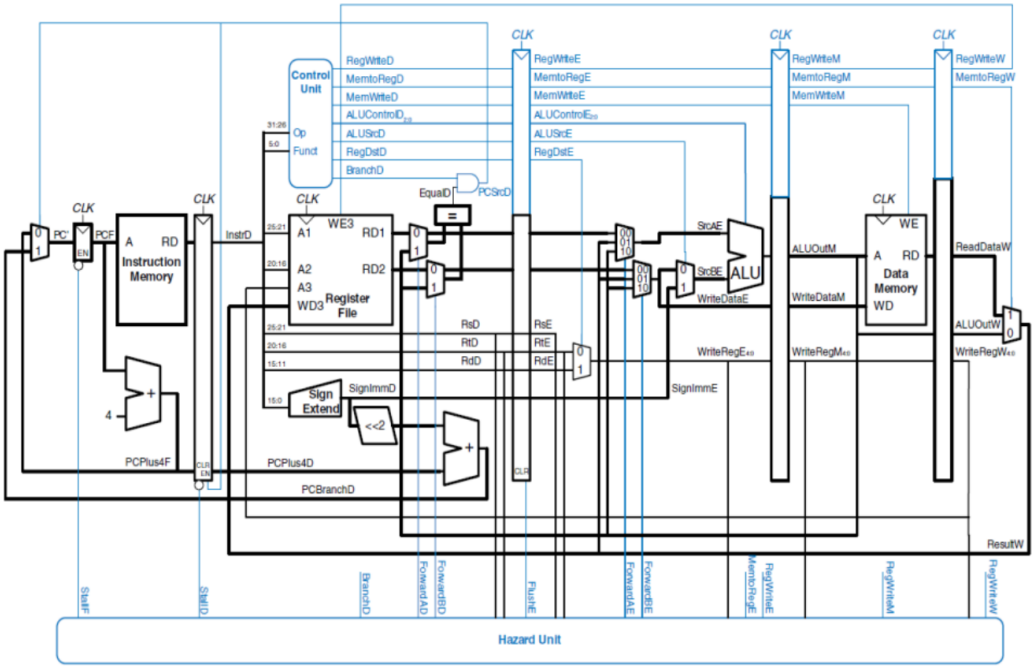
lw	addu/subu	M-E
		W-E
			暂停
			转发
				lw $5,4($0)
				instr(0,1)
				addu $6,$5,$4@

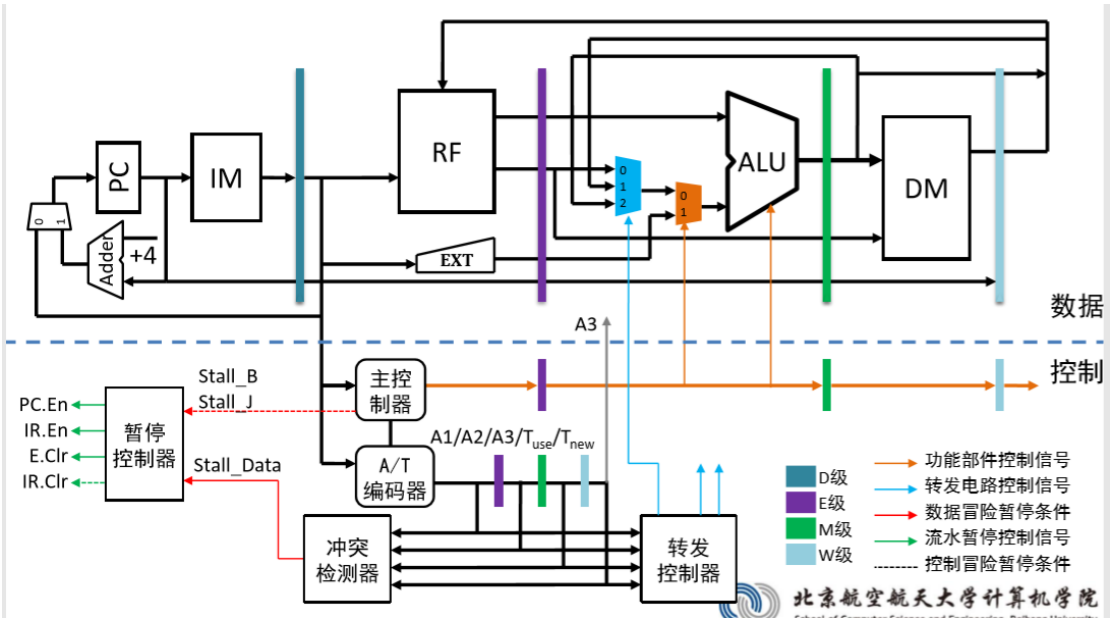
	ori	M-E W-E	暂停 转发	lw $5,4($0) instr(0,1) ori $6,$5,$4
	beq	E-D M-D W-D	暂停 暂停 转发	lw $5,4($0) instr(0,1,2) beq $5,$4,loop@
	jr	E-D	暂停	lw $5,4($0)
		M-D W-D	暂停 转发	instr(0,1,2) jr $5
	sw	W-M	转发	lw $5,4($0) sw $5,0($5)

	lw	M-E
		W-E
			转发
			转发
				lw $5,0($0)
				instr(0,1)
				lw $6,0($5)

jal	addu/subu	M-E W-E	转发 转发	jal loop instr(0,1) addu $6,$31,$4@
	ori	M-E W-E	转发 转发	jal loop instr(0,1) ori $6,$31,$4
	beq	E-D M-D W-D	转发 转发 转发	jal loop instr(0,1,2) beq $5,$31,loop@
	jr	E-D M-D W-D	转发 转发 转发	jal loop instr(0,1,2) jr $31
	sw	W-M	转发	lw $5,4($0) sw $5,0($5)

	lw	M-E
		W-E
			转发
			转发
				lw $5,0($0)
				instr(0,1)
				lw $6,0($5)


CLK
Control Unit
RegWriteD
MemtoRegD
MemWriteD
ALUControlD
ALUSrcD
RegDstD
BranchD
Op
Funct
EqualD
PCSrcD
PC'
PCF
EN
Instruction Memory
A
RD
InstrD
WE3
A1
A2
A3
WD3
Register File
RD1
RD2
Sign Extend
SignImmD
<<2
PCPlus4F
PCPlus4D
PCBranchD
RsD
RtD
RdD
RsE
RtE
RdE
SignImmE
SrcAE
SrcBE
ALU
WriteDataE
WriteRegE
RegWriteE
MemtoRegE
MemWriteE
ALUControlE
ALUSrcE
RegDstE
ALUOutM
WriteDataM
WriteRegM
RegWriteM
MemtoRegM
MemWriteM
WE
Data Memory
WD
ReadDataW
ALUOutW
WriteRegW
RegWriteW
MemtoRegW
ResultW
StallF
StallD
BranchD
ForwardAD
ForwardBD
FlushE
ForwardAE
ForwardBE
Hazard Unit


PC
IM
Adder
+4
RF
EXT
ALU
DM
数据
控制
A3
Stall_B
Stall_J
Stall_Data
PC.En
IR.En
E.Clr
IR.Clr
暂停控制器
主控制器
A/T编码器
A1/A2/A3/Tuse/Tnew
冲突检测器
转发控制器
D级
E级
M级
W级
功能部件控制信号
转发电路控制信号
数据冒险暂停条件
流水暂停控制信号
控制冒险暂停条件
北京航空航天大学计算机学院

构造指令集的T_{use}

- 思路：结合流水线架构，逐条指令构造T_{use}及T_{new}
- T_{use}注意事项
  - 1）只关注每条指令的操作语义
  - 2）指令可能有2个不同的T_{use}，如sw
  - 3）指令集或流水线架构的变化，均可能导致T_{use} 变化
    - 例如：流水线从5级变为6级且第5级为访存，则sw的rt将会延后1级被使用，故rt的T_{use}会变为{0,1,2,**3**}

	T_{use}	
	rs	rt
add	1	1
sub	1	1
andi	1	
ori	1	
lw	1	
sw	1	2
beq	0	0
jr	0	
	{0,1}	{0,1,2}

构造指令集的T_{new}

- 思路：结合流水线架构，逐条指令构造T_{use}及T_{new}
- T_{new}注意事项：
  - 一旦减为0，则不再继续减少！
    - 0：有效结果已经产生了
    - 非0：有效结果尚未产生
- 为了便于分析，用产生结果的功能部件来代表指令
  - 例如，ALU可以代表所有的计算类指令

指令	功能部件	T_{new}		
		E	M	W
add	ALU	1	0	0
sub	ALU	1	0	0
andi	ALU	1	0	0
ori	ALU	1	0	0
lw	DM	2	1	0
sw				
beq				
jal	PC	0	0	0

产生结果的功能部件

E			M			W		
ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
1	2	0	0	1	0	0	0	0

根据策略矩阵构造暂停条件的一般性方法

rs策略矩阵

T_{use} \ T_{new}	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F

```
Stall_RS0_e1 = . . .
Stall_RS0_e2 = . . .
Stall_RS0_m1 = . . .
Stall_RS1_e2 = . . .

Stall_RS = Stall_RS0_e1 |
           Stall_RS0_e2 |
           . . .
```

+

rt策略矩阵

T_{use} \ T_{new}	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F	F	F
1	F	S	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F

```
Stall_RT0_e1 = . . .
Stall_RT0_e2 = . . .
Stall_RT0_m1 = . . .
Stall_RT1_e2 = . . .

Stall_RT = Stall_RT0_e1 |
           Stall_RT0_e2 |
           . . .
```

↓

```
Stall = Stall_RS |
        Stall_RT
```

Q:
如何表示T_{new}与T_{use}?

策略矩阵制导构造暂停案例

- 当建立策略矩阵后，可以反向构造暂停和转发案例
  - 示例：rs策略矩阵制导的rs相关暂停用例

rs策略矩阵

T_{use} \ T_{new}	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S1	S2	F	F	S3	F	F	F	F
1	F	S4	F	F	F	F	F	F	F

TIPS
load类、store类、运算类、b类均包含多条指令。
因此，从指令角度，造成暂停的指令组合数量巨大。

rs相关暂停用例

S1	运算类 **$1**, $x, $y b类 **$1**, $x, im	运算类 **$1**, $x, $y jr **$1**	
S2	load类 **$1**, x($y) b类 **$1**, $x, im	load类 **$1**, x($y) jr **$1**	
S3	load类 **$1**, x($y) XXXXX b类 **$1**, $x, im	load类 **$1**, x($y) XXXXX jr **$1**	
S4	load类 **$1**, x($y) 运算类 $x, **$1**, $y	load类 **$1**, x($y) load类 $x, y(**$1**)	load类 **$1**, x($y) store类 $x, x(**$1**)

策略矩阵制导构造转发案例

- 示例：rt策略矩阵制导的可以转发的rt相关用例

rt策略矩阵

T_{use} \ T_{new}	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
0	S	S	F	F	S	F	F3	F3	F3
1	F	S	F	F	F	F	F3	F3	F3
2	F	F1	F	F	F2	F	F3	F3	F3

TIPS
相对于转发类相关数据更为惊人。
覆盖性分析方法带来很多重要启示，进一步深化流水线认识。

rt相关转发用例

F1	load类 **$1**, $x, $y store类 **$1**, x($y)	启示：W级应该有向M级的转发通路
F2	load类 **$1**, x($y) XXXXX store类 **$1**, x($y)	启示：同步信息的流水线寄存器也是需求点
F3	运算类 **$1**, $x, $y XXXXX XXXXX store类 **$1**, x($y)	启示：RF需要支撑内部转发（2017新方法也可以采用外部显式转发）

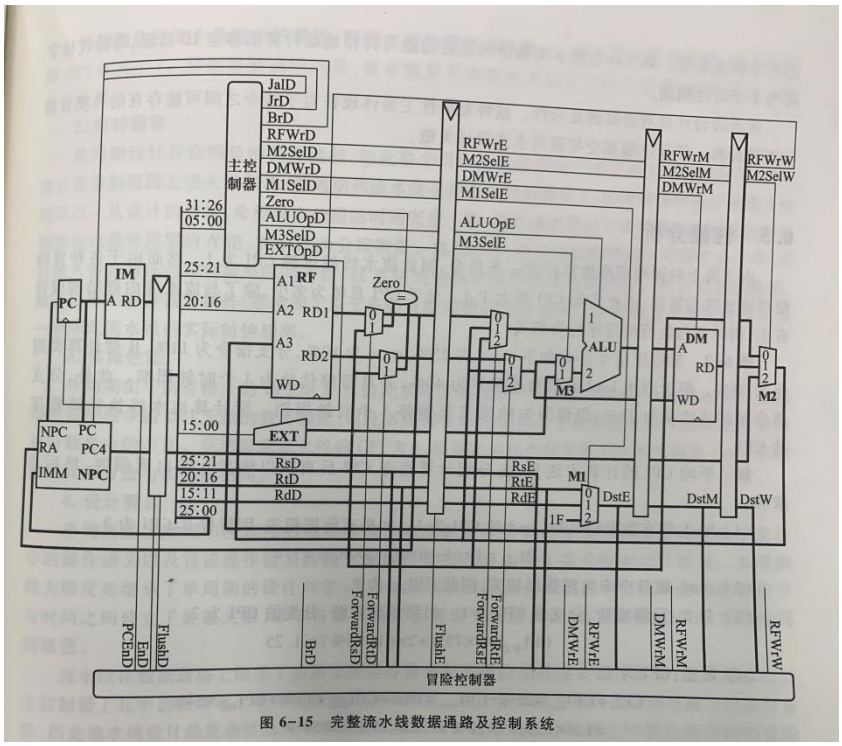


图 6-15 完整流水线数据通路及控制系统

		jr	jal	j	lui	beq	sw	lw	addu	subu	ori	MUX	0	1	2	3
PC		RF-RD1	NPC	NPC	ADD4	ADD4|NPC	ADD4	ADD4	ADD4	ADD4	ADD4	MUX-PC	ADD4	NPC	RF-RD1	
IM		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC				
ADD4		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC				
D级	IR-D	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM				
	PC4-D	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4				
RF	A1	IR-D(rs)			IR-D(rs)	IR-D(rs)	IR-D(rs)	IR-D(rs)	IR-D(rs)	IR-D(rs)	IR-D(rs)	IR-D(rs)				
	A2					IR-D(rt)	IR-D(rt)		IR-D(rt)	IR-D(rt)		IR-D(rt)				
EXT					IR-D(imm)		IR-D(imm)	IR-D(imm)			IR-D(imm)	IR-D(imm)				
	PC4		PC4-D	PC4-D		PC4-D						PC4-D				
NPC(转移+4)	IM26		IR-D(imm)	IR-D(imm)		IR-D(imm)						IR-D(imm)				
	zero					CMP-zero						CMP-zero				
CMP	D1					RF-RD1						RF-RD1				
	D2					RF-RD2						RF-RD2				
E级	V1-E				RF-RD1		RF-RD1	RF-RD1	RF-RD1	RF-RD1	RF-RD1	RF-RD1				
	V2-E						RF-RD2		RF-RD2	RF-RD2		RF-RD2				
	A1-E						IR-D(rs)	IR-D(rs)	IR-D(rs)	IR-D(rs)	IR-D(rs)	IR-D(rs)				
	A2-E						IR-D(rt)		IR-D(rt)	IR-D(rt)		IR-D(rt)				
	A3-E		31		IR-D(rt)			IR-D(rt)	IR-D(rd)	IR-D(rd)	IR-D(rd)	MUX-A3	IR-D(rt)	IR-D(rd)	31	
	E32-E				EXT-out		EXT-out	EXT-out			EXT-out	EXT-out				
	PC4-E		PC4-D									PC4-D				
ALU	A				V1-E		V1-E	V1-E	V1-E	V1-E	V1-E	V1-E				
	B				E32-E		E32-E	E32-E	V2-E	V2-E	E32-E	MUX-ALUB	V2-E	E32-E		
M级	V2-M						V2-E					V2-E				
	A2-M						A2-E					A2-E				
	AO-M				ALU-out		ALU-out	ALU-out	ALU-out	ALU-out	ALU-out	ALU-out				
	A3-M		A3-E		A3-E			A3-E	A3-E	A3-E	A3-E	A3-E				
	PC4-M		PC4-E									PC4-E				
DM	A						AO-M	AO-M	AO-M	AO-M	AO-M	AO-M				
	WD						V2-M					V2-M				
W级	A3-W		A3-M		A3-M			A3-M	A3-M	A3-M	A3-M	A3-M				
	PC4-W		PC4-M									PC4-M				
	AO-W				AO-M			AO-M	AO-M	AO-M	AO-M	AO-M				
	DR-W							DM-out				DM-out				

	Tnew	功能部件
add/addu/sub/subu/and/or/xor/nor sll/sra/sll/mflo/mfhi addi/addiu/andi/ori/xori/lui/ sllv/srlv/srav/slt/sltu/slti/sltiu	1	ALU
lbu/lb/lhu/lh/lw	2	DM
jal/jalr	0	PC

	Tuse	
	rs	rt
beq/bne	0	0
blez/bgez/bltz/bgtz/jr/jalr	0	
add/addu/sub/subu/and/or/xor/nor sllv/srlv/srav/slt/sltu/mult/div/multu/divu	1	1
sb/sh/sw	1	2
sll/sra/sll		1
addi/addiu/andi/ori/xori/lui/ lbu/lb/lhu/lh/lw/slti/sltiu/mtlo/mthi	1	

将指令分为 7 类，每一类的数据通路大致相同，从而按照类来进行冲突分析。

addu/add/subu/sub/nor/or/xor/slt/sltu/sllv/srav/srlv/and	1
addi/addiu/andi/lui/ori/xori/slti/sltiu/	2
lb/lbu/lh/lhu/lw	3
sb/sw/sh	4
sll/sra/srl	5
b	6
j	7