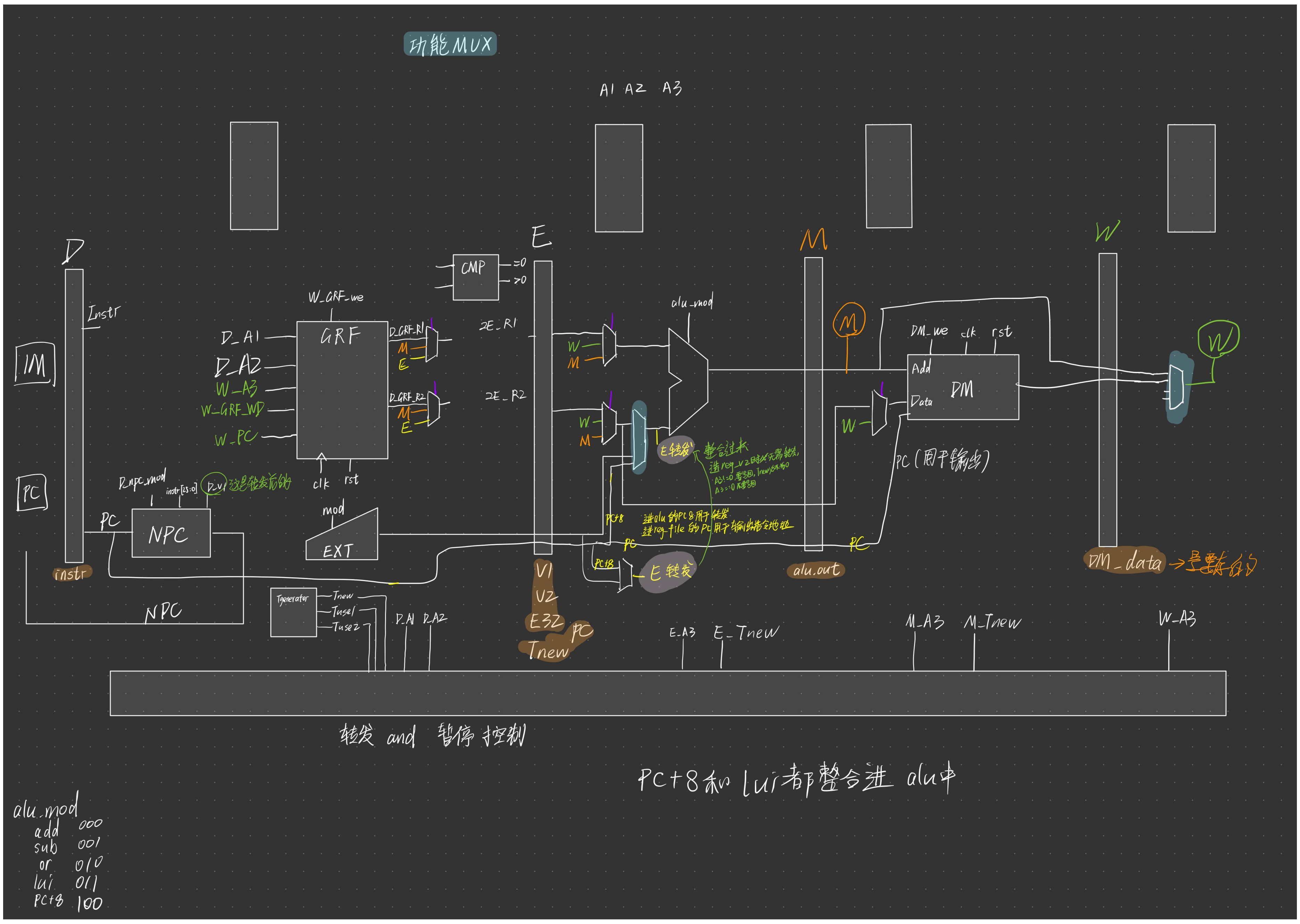
P5设计文档

顶层设计



功能MUX
A1 A2 A3
D
E
M
W
IM
PC
Instr
D_A1
D_A2
W_A3
W_GRF_WD
W_PC
W_GRF_we
GRF
clk
rst
CMP
=0
>0
D_GRF_R1
D_GRF_R2
2E_R1
2E_R2
D_npc_mod
instr[25:0]
D_V1 这是转发后的
NPC
mod
EXT
instr
Tgenerator
Tnew
Tuse1
Tuse2
D_A1 D_A2
V1
V2
E32
PC
Tnew
alu_mod
E转发
整合过来
进reg_v2时也无需转发,
A3!=0要写回, Tnew必不为0
A3==0不要写回
PC+8
进alu的PC8用于转发
进reg_file的PC用于输出指令地址
PC
PC+8
E转发
alu.out
DM_we
clk
rst
Add
Data
DM
PC(用于输出)
DM_data → 寻数的
E_A3
E_Tnew
M_A3
M_Tnew
W_A3
转发 and 暂停 控制
PC+8和lui都整合进 alu中
alu_mod
add 000
sub 001
or 010
lui 011
PC+8 100

设计：

Datapath：

I, D, E, M, W 5模块　　转发MUX　　内含于

ID　EE　IM　IW　4个流水寄存器

→ 允许有多的无用寄存器
不考虑性能，仅考虑简单+功能

Control：

分布式指令译码（内含跳转控制信号，PC+8的问题）

D级有自己的AT控制器，其它级AT-1即可

冒险控制信号（Stall and bypass）

编码注意：

用个宏头文件去了来写宏定义，如果仅是一个.v中用的话localparameter也行

→ 仅local用就不要用宏了，避免重名

命名上：流水级_元件_方向(i/o)（定义写在使用的后面ok吗？）

addu　subu
ori　Lui
lw　sw
beq　j　jal　jr
nop

重要机制的实现

转发

转发

若无需读数据，如何处理？⇒ A1 A2 置0，0寄存器必不转发

无需处理是否应转发，直接转发即可
若是要读reg的情况，满足条件当然转
若是无需读reg的情况，后面还一个MUX，转了不影响（此时默认转发MUX信号为0）

优选新的那一条指令

转发条件：
编号相同 且不为0（可能有错误指令）
已产生正确结果 Tnew=0
~~grf_we=1~~ 不需要

D级：需分析与E M W(→GRF内部转发)级的数据转发问题

A1：
```
D_A1==M_A3 && M_A3!=5'd0 && M_Tnew==0
D_A1==E_A3 && E_A3!=5'd0 && E_Tnew==0
```
```
D_mux_A1_mod=2'd1
D_mux_A1_mod=2'd2
} else=0
```

A2：
```
D_A2==M_A3 && M_A3!=5'd0 && M_Tnew==0
```
如上A1
```
D_mux_A2_mod=1 else=0
```

E级：仅需分析与M W级的数据转发问题

A1：
```
E_A1==W_A3 && W_A3!=5'd0 && W_Tnew必为0
E_A1==M_A3 && M_A3!=5'd0 && M_Tnew==2'd0
```
```
E_mux_A1_mod=2'd1
E_mux_A1_mod=2'd2
```

A2与A1一样：

M级：仅需分析与W级的数据转发问题

A2：
```
M_A2==W_A3 && W_A3!=5'd0
```

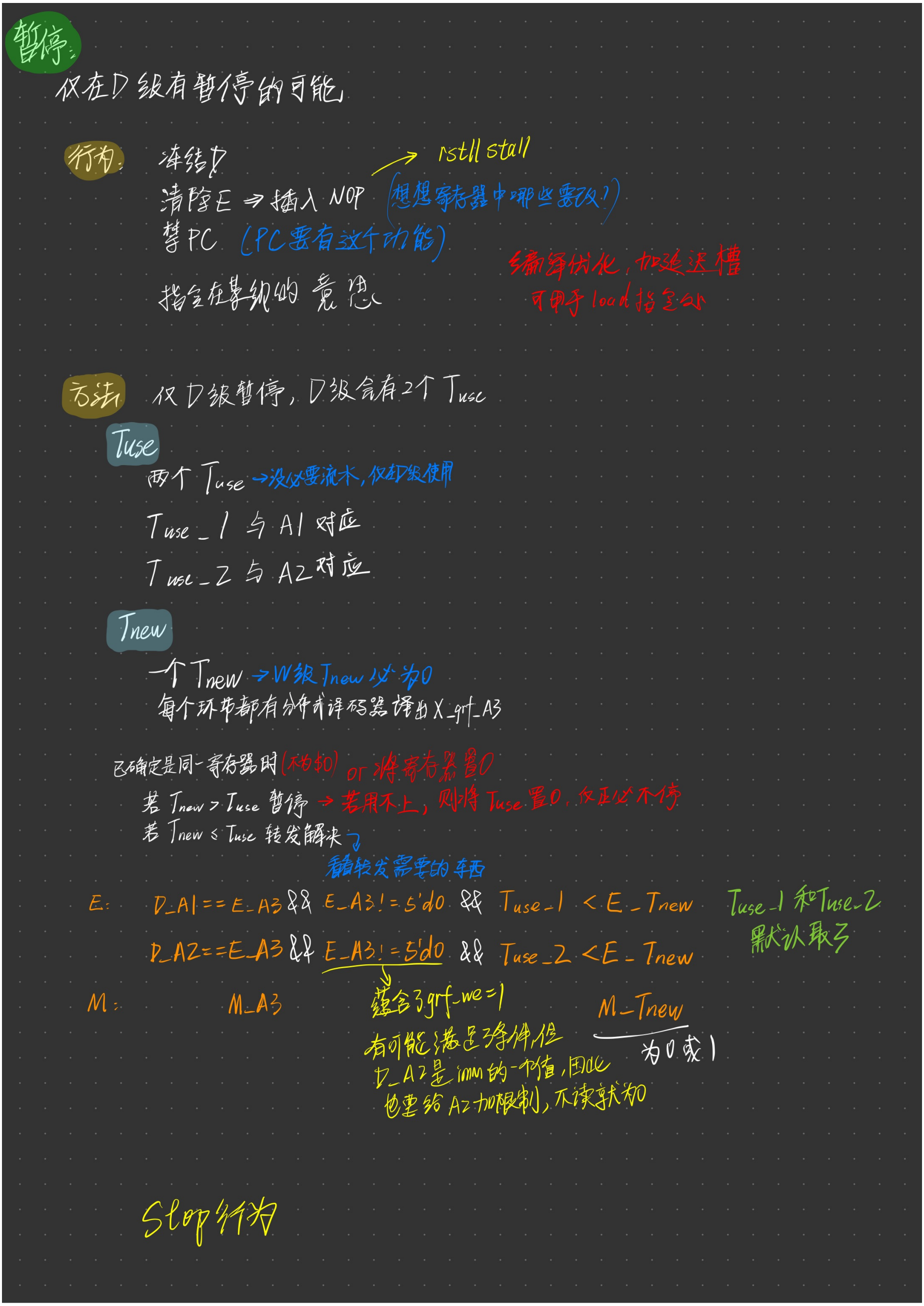
跳转：

用了延迟槽，base变为pc+8
jal，jalR的回写都视为普通的回写来做，

control多一个输出信号：is_branch

beq	pc+8	or	PC+8+□<<2
jr	reg		
jal	P		
j			

暂停


暂停
仅在D级有暂停的可能
行为
冻结D
清除E ⇒ 插入 NOP
禁PC (PC要有这个功能)
stall
(想想寄存器中哪些要改?)
指令在某级的意思
编译优化，加延迟槽
可用于load指令处
方法
仅D级暂停，D级会有2个Tuse
Tuse
两个Tuse → 没必要流水，仅在D级使用
Tuse_1 与 A1 对应
Tuse_2 与 A2 对应
Tnew
一个Tnew → W级Tnew必为0
每个环节都有分析式译码器译出X_grf_A3
已确定是同一寄存器时 (不为$0) or 将寄存器置0
若 Tnew > Tuse 暂停 → 若用不上，则将Tuse置0，仅正必不停
若 Tnew ≤ Tuse 转发解决
看看转发需要的东西
E: D_A1==E_A3 && E_A3!=5'd0 && Tuse_1 < E_Tnew
D_A2==E_A3 && E_A3!=5'd0 && Tuse_2 < E_Tnew
Tuse_1 和Tuse_2 默认取3
M: M_A3
蕴含了grf_we=1
M_Tnew
为0或1
有可能满足3条件，但
D_A2是imm的一个值，因此
也要给A2加限制，不读就为0
Stop行为

数据通路

			addu	subu	ori	lui	lw	sw	beq	j	jal	jr	nop
D	grf	a1	D_a1	D_a1	D_a1		D_a1	D_a1	D_a1			D_a1	
		a2	D_a2	D_a2				D_a2	D_a2				
		a3	W_a3	W_a3	W_a3	W_a3	W_a3				W_a3=31		
		wd	W_wd	W_wd	W_wd	W_wd	W_wd				W_wd		
		we	W_grf_we	W_grf_we	W_grf_we	W_grf_we	W_grf_we				W_grf_we		
		clk											
		rst											
	cmp	cmp_in_1							D_v1				
		cmp_in_2							D_v2				
	ext	ext_i			D_imm	D_imm	D_imm	D_imm					
		mod			ext_mod(00)	ext_mod(10)	ext_mod(01)	ext_mod(01)					
	npc	instr[25:0]							instr[25:0]	instr[25:0]	instr[25:0]		
		grf_v										D_v1	
		npc_pc_in	pc	pc	pc	pc	pc	pc	pc	pc	pc	pc	
		npc_mod	npc_mod(000)	npc_mod(000)	npc_mod(000)	npc_mod(000)	npc_mod(000)	npc_mod(000)	npc_mod(000) npc_mod(001)	npc_mod(010)	npc_mod(010)	npc_mod(011)	
E	alu	alu_in_1	E_v1	E_v1	E_v1		E_v1						
		alu_in_2	E_v2	E_v2	E_v2	E_ext_o	E_ext_o	E_ext_o			pc 8(流水pc)		
		alu_mod	`alu_add(000)	`alu_sub(001)	`alu_or(010)	`alu_in2(011)	`alu_add(000)	`alu_add(000)			`alu_in2(011)		
M	DM	add					M_alu_o	M_alu_o					
		data						M_v2					
		DM_we						M_DM_we					
		clk											
		rst											
W	只有一个mux, 用于选出W_wd, 写入数据		W_alu_o	W_alu_o	W_alu_o	W_alu_o	M_DM_o				M_alu_o		

流水寄存器	
instr	DEMW
pc	DEMW
v1	E
v2	EM
e32	E
alu_out	MW
DM_data	W
Tnew	EM

pc8用于转发，pc用于输出DM中的pc结果

e32和pc再E级需要一个mux来选择到底是在E级要转发哪一个（pc要加8），随后e32和pc8就直接进入alu中，后续作为alu_o转发

信号设计

AT设计

lui可以在E级转发而没有转发

转发时会对A3进行判断，如果A3为0则不进行转发。无论Tnew

	addu	subu	ori	lui	lw	sw	beq	j	jal	jr	nop
Tnew	1	1	1	0	2	0（默认）	0（默认）	0（默认）	0	0（默认）	0（默认）
T_use1	1	1	1	3（默认）	1	1	0	3（默认）	3（默认）	0	3（默认）
T_use2	1	1	3（默认）	3（默认）	3（默认）	2	0	3（默认）	3（默认）	3（默认）	3（默认）

Tues越小越会暂停，默认取最大的3

Tnew越大越会暂停，默认取最小值0

Tuse<Tnew

控制信号

	addu	subu	ori	lui	lw	sw	beq	j	jal	jr	nop
grf_a1	aa1[25:21]	aa1	aa1		aa1	aa1	aa1			aa1	
grf_a2	aa2[20:16]	aa2				aa2	aa2				
grf_a3	aa3[15:11]	aa3	aa2	aa2	aa2	0	0	0	31	0	0
grf_we	1	1	1	1	1				1		
ext_mod			`ext_0(00)	`ext_lui(10)	`ext_sign(01)	`ext_sign(01)					
npc_mod	`npc_add4 (000)	`npc_add4 (000)	`npc_add4 (000)	`npc_add4 (000)	`npc_add4 (000)	`npc_add4 (000)	`npc_add4 (000) `npc_brch (001)	`npc_j (010)	`npc_j (010)	`npc_jr (011)	
alu_mod	`alu_add(000)	`alu_sub(001)	`alu_or (010)	`alu_in2 (011)	`alu_add(000)	`alu_add(000)			`alu_in2(011)(输入pc8)		
mux_alu_in2	000 (v2)	000 (v2)	001 (ext_out)	001 (ext_out)	001 (ext_out)	001 (ext_out)			010 (pc8)		
DM_we						1					
mux_W_wd	00 (alu_out)	00 (alu_out)	00 (alu_out)	00 (alu_out)	01 (DM_out)				00 (alu_out)		

`alu_add(000)
`alu_sub (001)
`alu_or (010)
`alu_in2 (011)

`ext_0(00)
`ext_sign(01)
`ext_lui(10)

`npc_add4 (000)
`npc_brch (001)
`npc_j (010)
`npc_jr (011)

直接将alu的第二个输入放到输出处，用于lui和jal。整合进alu之后M级的转发、W级的写入就很方便了
转发时会对A3进行判断，如果A3为0则不进行转发。无论Tnew，A3默认为0而不是instr[15:11]

P5思考题

流水线冒险

Q1:在采用本节所述的控制冒险处理方式下，PC 的值应当如何被更新？请从数据通路和控制信号两方面进行说明。

```
对于顺序处理，PC+4。
对于跳转指令，
    对于brch类，将base置为PC+4再进行操作。
    对于 j  类，直接使用位拼接运算符计算
```

Q2:对于 jal 等需要将指令地址写入寄存器的指令，为什么需要回写 PC+8 ？

```
编译时会对指令进行优化，保证跳转指令后的一条指令可以直接顺序执行，无需考虑当前跳转指令的情况。
因此，当使用jr跳回时应该跳过这一条指令，因此需要写回PC+8
```

数据冒险的分析

Q：为什么所有的供给者都是存储了上一级传来的各种数据的**流水级寄存器**，而不是由 ALU 或者 DM 等部件来提供数据?

```
1. 使用流水寄存器的优点：在一个时间周期内，转发回去的数据都是稳定的。相比之下如果连接部件，可将电路视为逻辑电路，其信号不稳定。
2. 原理上看直接连接部件来转发是可行的，但是这样意味着需要延长时钟周期的长度。例如从M级转发回E级，需要等M级处理完以后将数据转发回ALU的输入端口让ALU再跑一次，此时单个时钟周期时长翻倍。得不偿失。
```

AT 法处理流水线数据冒险

Q1: think1-4

1. **Thinking 1**：如果不采用已经转发过的数据，而采用上一级中的原始数据，会出现怎样的问题？试列举指令序列说明这个问题。

```
例如
    add $t1, $t1, $t2
    add $t3, $t1, $t2
第一条add还没有将结果写入寄存器中，add就已经读了$t1
```

2. **Thinking 2**：我们为什么要对 GPR 采用内部转发机制？如果不采用内部转发机制，我们要怎样才能解决这种情况下的转发需求呢?

```
采用内部转发机制是因为GPR本身既可以视为是D级部件，也可以视为W级部件，而W级行为可能会对D级行为产生影响，使用内部转发可以消除这种影响，使得读取GRF值的时候可以读出正确的值，是对GRF的优化。
如果不采用内部转发，需要在寄存器外部放置一个MUX来对"GRF输出信号"和"GRF写入信号"进行选择，选择方式与内部转发方法一致。    或者，在发生GRF冲突时，强制将流水线暂停一个周期。
```

3. **Thinking 3**：为什么 0 号寄存器需要特殊处理？选择信号的生成规则是：只要**当前位点的读取寄存器地址和某转发输入来源的写入寄存器地址相等且不为 0**

```
原因有两个：
1. 0号寄存器无需转发，可以直接读取值0
2. 利用0号寄存器不可写入的特点，对于无需写入的情况，可以直接将A3置为0，表示其无需转发。
```

4. **Thinking 4**：什么是“最新产生的数据”？

```
对E级而言，M级和W级都有可能需要向E级进行转发，此时应该选择“最新”的结果即M级的信息来转发。
```

Q2： 在 AT 方法讨论转发条件的时候，只提到了“供给者需求者的A相同，且不为 0”，但在 CPU 写入 GRF 的时候，是有一个 we 信号来控制是否要写入的。为何在 AT 方法中不需要特判 we 呢？为了**用且仅用** A 和 T 完成转发，在翻译出 A 的时候，要结合 we 做什么操作呢?

```
AT法中，A表示需要写入的寄存器编号。对0号寄存器，无需转发，其值一定为0。对于不需要写入寄存器的指令而言，可以将A3置为0，表示无需转发。此时，只要A!=0，说明其必然是需要写入寄存器的，x_grf_we必定为1。    在翻译A时，如果grf_we不为0，则置A为0。
```