说明

处理器应支持如下指令集：{ add, sub, ori, lw, sw, beq, lui, jal, jr, nop }

cal_R:add,sub

cal_I:ori

load:lw

store:sw

B类:beq

J类:jal

特殊:jr

lui

功能模块

F级：

IFU(取指令单元)

信号名	方向	位宽	描述
clk	I	1	时钟信号
reset	I	1	同步复位信号
en	I	1	使能信号
npc	I	32	下一条指令地址
pc	O	32	当前指令地址
instr	O	32	当前指令

D级：

EXT

信号名	方向	位宽	描述
EXT_op	I	1	lui拓展(10)/符号扩展(01)/零扩展(00)
in	I	16	16位立即数
out	O	32	扩展后32位结果

NPC(下一指令计算单元)

信号名	方向	位宽	描述

信号名	方向	位宽	描述
pc	I	32	F级pc
NPC_op	I	3	指令类型选择: 000:顺序执行 001:B类/beq 010:jal 011:jr
instr_offset_ext	I	32	D级16位立即数零扩展/符号扩展的32位
instr_index	I	26	D级指令数据的0:25，用于计算jal所要跳转的地址
pc_rs	I	32	D级指令数据的21:25所表示rs寄存器中存储的地址
judge_b	I	1	D级b类跳转指令是否满足 1:跳 2:不跳
npc	O	32	下一条要被执行的指令的地址

GRF(通用寄存器组)

信号名	方向	位宽	描述
clk	I	1	时钟信号
reset	I	1	同步复位信号 1：复位信号有效 0：复位信号无效
pc	I	32	当前指令地址
WE	I	1	写使能信号 1：写入有效 0：写入失效
WA	I	5	地址输入信号，指定32个寄存器中的一个，将其作为写入目标
WD	I	32	写入的数据
RA1	I	5	地址输入信号，指定32个寄存器中的一个，将其中的数据读出到RD1
RA2	I	5	地址输入信号，指定32个寄存器中的一个，将其中的数据读出到RD2
RD1	O	32	输出A1指定的寄存器中的32位数据
RD2	O	32	输出A2指定的寄存器中的32位数据

CMP(B类指令比较单元)

信号名	方向	位宽	描述
RD1	I	32	输入CMP单元的第一个数据
RD2	I	32	输入CMP单元的第二个数据
CMP_op	I	3	CMPOp功能选择信号 0x000：beq判断 (可扩展)
out	O	1	判断结果输出 1: 判断结果为真 0：判断结果为假

E级：

ALU(逻辑运算单元)

信号名	方向	位宽	描述
A	I	32	第一个运算数
B	I	32	第二个运算数
ALU_op	I	3	选择运算方式(**扩展要点**)
ALU_result	O	32	运算结果

M级：

DM(数据存储器)

信号名	方向	位宽	描述
clk	I	1	时钟信号
reset	I	1	同步复位信号 1：复位信号有效 0：复位信号无效
pc	I	32	当前指令地址
WE	I	1	写使能信号 1：写入有效 0：写入失效
WA	I	12	内存地址输入信号
WD	I	32	数据输入信号
RD	O	32	输出A指定的内存中的32位数据

流水寄存器

Tnew也在流水寄存器中

DREG(IF/ID流水寄存器)

方向	信号名	位宽	描述	输入来源
I	clk	1	时钟信号	mips.v中的clk
I	reset	1	同步复位	
I	en	1	D级寄存器使能信号	？？？
I	F_instr	32	F级instr输入	IFU_instr
I	F_pc	32	F级pc输入	IFU_pc
O	D_instr	32	D级instr输出	
O	D_pc	32	D级pc输出	

EREG(ID/EXE寄存器)

方向	信号名	位宽	描述	输入来源
I	clk	1	时钟信号	mips.v中的clk
I	reset	1	同步复位	
I	clr	1	E级寄存器清空信号	HCU中stall信号
I	D_instr	32	D级instr输入	
I	D_pc	32	D级pc输入	
I	D_GRF_RD1	32	D级GRF_RD1输入	转发
I	D_GRF_RD2	32	D级GRF_RD2输入	转发
I	D_GRF_WA	5	D级GRF_WA输入	MUX
I	D_EXT_out	32	D级EXT_out输入	通过EXT模块扩展出的数据
I	Tnew_D	2	D级指令的Tnew	DMCU产生
O	E_instr	32	E级instr输出	
O	E_pc	32	E级pc输出	
O	E_GRF_RD1	32	E级GRF_RD1输出	
O	E_GRF_RD2	32	E级GRF_RD2输出	
O	E_GRF_WA	5	E级GRF_WA输出	
O	E_EXT_out	32	E级EXT_out输出	
O	Tnew_E	2	max{Tnew_D - 1,0}	

MREG(EX/MEM流水寄存器)

方向	信号名	位宽	描述	输入来源
I	clk	1	时钟信号	mips.v中的clk
I	reset	1	同步复位	
I	E_instr	32	E级instr输入	
I	E_pc	32	E级pc输入	
I	E_GRF_RD2	32	E级GRF_RD2输入	////
I	E_GRF_WA	5	E级GRF_WA输入	MUX
I	E_ALU_result	32	E级ALU_result输入	
I	Tnew_E	2	E级指令的Tnew	

方向	信号名	位宽	描述	输入来源
O	M_instr	32	M级instr输出	
O	M_pc	32	M级pc输出	
O	M_GRF_RD2	32	M级GRF_RD2输出	////
O	M_GRF_WA	5	M级GRF_WA输出	
O	M_ALU_result	32	M级ALU_result输出	
O	Tnew_M	2	max{Tnew_E - 1,0}	

WREG(MEM/WB流水寄存器)

方向	信号名	位宽	描述	输入来源
I	clk	1	时钟信号	mips.v中的clk
I	reset	1	同步复位	
I	M_instr	32	M级instr输出	
I	M_pc	32	M级pc输入	
I	M_DM_RD	32	M级DM_RD输入	
I	M_GRF_WA	5	M级GRF_WA输入	
I	M_ALU_result	32	M级ALU_result输入	
O	W_instr	32	W级instr输出	
O	W_pc	32	W级pc输出	
O	W_DM_RD	32	W级DM_RD输出	
O	W_GRF_WA	5	W级GRF_WA输出	
O	W_ALU_result	32	W级ALU_result输出	

控制模块

MCU（主控制器模块）

- 输入

信号名	位宽	描述
pc	32	当前pc
instr	32	当前instr

- 输出

信号名	位宽	描述
Op	6	instr[31:26]
Func	6	instr[5:0]
rs	5	instr[25:21]
rt	5	instr[20:16]
rd	5	instr[15:11]
instr_index	26	instr[25:0]
Imm	16	instr[15:0]
RegWrite	1	Reg写数据使能
Sel_GRF_WA	2	寄存器堆写入端地址选择
Sel_GRF_WD	2	寄存器堆写入端数据源选择
Sel_ALU_B	1	ALU输入端B数据源选择
Sel_E_out	1	E级储存的计算结果选择
Sel_M_out	1	M级储存的计算结果选择
Sel_W_out	2	W级储存的计算结果选择
MemRead	1	DM读数据使能
MemWrite	1	DM写数据使能
ALU_op	3	ALU模块功能选择信号
NPC_op	3	指令类型选择: 000:顺序执行 001:B类 010:jal 011:jr
CMP_op	3	CMP模块功能选择信号 0x000：beq判断 (可扩展)
EXT_op	2	EXT模块功能选择信号 lui(10)/符号扩展(01)/零扩展(00)
Tuse_rs	2	具体值见转发与暂停
Tuse_rt	2	同上
Tnew_D	2	同上

HCU（冒险控制器模块）

- **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**
- **写使能信号有效**

根据以上条件我们可以生成上面的5个HMUX选择信号，选择信号的输出值应遵循“就近原则”，及最先产生的数据最先被转发。

- 输入

信号名	位宽	描述
D_GRF_RA1	5	D级GRF_RA1输入
D_GRF_RA2	5	D级GRF_RA2输入
E_GRF_RA1	5	E级GRF_RA1输入
E_GRF_RA2	5	E级GRF_RA2输入
E_GRF_WA	5	E级GRF_WA输入
E_WE	1	E级写使能信号
M_GRF_RA2	5	M级GRF_RA2输入
M_GRF_WA	5	M级GRF_WA输入
M_WE	1	M级写使能信号
W_GRF_WA	5	W级GRF_WA输入
W_WE	1	W级写使能信号
Tuse_rs	2	D级MCU中输出的Tuse_rs信号
Tuse_rt	2	D级MCU中输出的Tuse_rt信号
Tnew_E	2	E级Tnew_E信号输入
Tnew_M	2	M级Tnew_E信号输入
Tnew_W	2	W级Tnew_E信号输入

- 输出

信号名	位宽	作用级	描述
FW_CMP_RD1_D	2	D	对HMUX_CMP_D1的输出进行选择
FW_CMP_RD2_D	2	D	对HMUX_CMP_D2的输出进行选择
FW_ALU_A_E	2	E	对HMUX_ALU_A的输出进行选择
FW_ALU_B_E	2	E	对HMUX_ALU_B的输出进行选择
FW_DM_RD_M	1	M	对HMUX_DM_RD的输出进行选择

信号名	位宽	作用级	描述
stall	1	F、D、M	暂停信号

选择器模块

功能MUX

内部选择数据输入

级别	MUX名	描述	控制信号	输出信号
D级	MUX_GRF_WA	00:D_instr_rd 01:D_instr_rt 10:0x1f	Sel_GRF_WA	GRF_WA
E级	MUX_ALU_B	0:E_GRF_RD2 1:E_ext	Sel_ALU_B	ALU_B
W级	MUX_GRF_WD	00:W_ALU_result 01:W_DM_RD 10:pc_8	Sel_GRF_WD	GRF_WD

流水级寄存器输出选择(数据通路中实现)

描述	控制信号	输出信号
选择E级存储的计算结果 0:E_EXT_out 1:E_pc_8	Sel_E_out	E_out
选择M级存储的计算结果 0:M_ALU_result 1:M_pc_8	Sel_M_out	M_out
选择W级存储的计算结果 0:W_ALU_result 1:W_DM_RD 2:W_pc_8	Sel_W_out	W_out

转发MUX

接收端口	MUX名	描述	控制信号	输出信号
CMP_RD1 NPC_pc_rs	HMUX_CMP_RD1_D	0:D_GRF_RD1 1:M_out 2:E_out	FW_CMP_RD1_D	D_GRF_RD1_f
CMP_RD2	HMUX_CMP_RD2_D	0:D_GRF_RD2 1:M_out 2:E_out	FW_CMP_RD2_D	D_GRF_RD2_f
ALU_A	HMUX_ALU_A_E	0:E_GRF_RD1 1:W_out 2:M_out	FW_ALU_A_E	E_GRF_RD1_f

接收端口	MUX名	描述	控制信号	输出信号
ALU_B	HMUX_ALU_B_E	0:E_GRF_RD2 1:W_out 2:M_out	FW_ALU_B_E	E_GRF_RD2_f
DM_WD	HMUX_DM_WD_M	0:M_GRF_RD2 1:W_out	FW_DM_WD_M	M_GRF_RD2_f

转发与暂停

- Tuse：指令进入 **D 级**后，其后的某个功能部件**再**经过多少时钟周期就**必须**要使用寄存器值。对于有两个操作数的指令，其**每个操作数的 Tuse 值可能不等**（如 store 型指令 rs、rt 的 Tuse 分别为 1 和 2 ）。
- Tnew：位于 **E 级及其后各级**的指令，再经过多少周期就能够产生要写入寄存器的结果。在我们目前的 CPU 中，W 级的指令Tnew 恒为 0；对于同一条指令，Tnew@M = max(Tnew@E - 1, 0)
- **Tuse表**

指令类型	Tuse_rs	Tuse_rt
calc_R	1	1
calc_I	1	X
load	1	X
store	1	2
branch	0	0
jump	X	X
jr	0	X

- **Tnew表**

产生结果的功能部件	指令类型	Tnew_D	Tnew_E	Tnew_M	Tnew_W
ALU	calc_R	2	1	0	0
ALU	calc_I	2	1	0	0
DM	load	3	2	1	0
	store	X	X	X	X
	branch	X	X	X	X
PC	jal	0	0	0	0
	jr	X	X	X	X
EXT	lui	1	0	0	0

然后我们Tnew和Tuse传入HCU（冒险控制器中），然后进行stall信号的计算。如果Tnew > TuseHCU中的stall信号值为1，此时执行以下操作——

- **冻结PC寄存器（IFU_en = ~stall = 0）**
- **冻结D级寄存器（D_en = ~stall = 0）**
- **清空E级寄存器（E_clr = stall = 1）**

思考题

1. **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。**

   MCU控制信号的生成

   可能还需要HCU中AT的更改、流水级寄存器

   - 寄存器立即数计算：`addi, addiu, slti, sltiu, andi, ori, xori, sll, srl, sra`

     EXT ALU
   - 寄存器寄存器计算：`add, addu, sub, subu, slt, sltu, and, or, nor, xor, sllv, srlv, srav`

     ALU
   - 根据寄存器分支：`beq, bne, bgez, bgtz, blez, bltz`

     CMP NPC EXT
   - 写内存：`sw, sh, sb`

     DM
   - 读内存：`lw, lh, lhu, lb, lbu`

     DM
   - 跳转并链接：`jal, jalr`

     NPC
   - 跳转寄存器：`jr, jalr`

     NPC
   - 加载高位：`lui`
   - 空指令：`nop`

2. **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。**

   译码方式：分布式译码+控制信号驱动型

   每一级都有一个MCU被实例化，仅有需要的端口被连接

   优势：较为灵活，“现译现用”有效降低了流水级间传递的信号量

   不足：需要实例化多个控制器，增加了后续流水级的逻辑复杂度

3. **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。**

   提前分支预判所需的数据需要从后续流水级转发而来，所以可能存在提前分支预判时正确数据并没有产生需要阻塞的情况(而不提前却能正常流水不用阻塞)

   ```
   1 add $1,$1,$2
   2 beq $1,$2,label
   ```

4. **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？**

jal 的延迟槽内的指令已经紧跟着 jal 指令运行了，不需要在返回的时候再次运行。

5. **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?**

   ①直接从功能部件转发会导致数据通路变长

   ②流水寄存器中存储的数据是上一级已经计算出来的数据，在当前周期内稳定输出。如果从功能部件提供转发数据，可能在正确转发数据生成前就转发了错误数据。

6. **我们为什么要使用 GPR 内部转发？该如何实现?**

   GPR 是一个特殊的部件，既可以视为 D 级的一个部件，也可以视为 W 级之后的流水线寄存器。也可以在外部实现HMUX。

```
assign RD1 = (WA == RA1 && WA && WE) ? WD : Regs [RA1];
assign RD2 = (WA == RA2 && WA && WE) ? WD : Regs [RA2];
```

7. **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?**

   需求者：CMP_RD1/NPC_pc_rs,CMP_RD2 EGRF ALU_A,ALU_B MGRF DM_WD

   供给者：E级流水寄存器：EXT_out,pc_8

   M级流水寄存器：ALU_result,pc_8

   W级流水寄存器：ALU_result,DM_RD,pc_8

   E→D M→D

   M→E W→E

   W→M

   需求者方有一个MUX，根据要用的RA与供给者方的RA、Tnew、WE选择

   供给者方有一个MUX，根据本级指令选择提供的数据