浙江大学
ZheJiang University

Computer Architecture Experiment

Jiang Xiaohong

Topics

- 0、Basic Knowledge
- 1、Warm up
- 2、simple 5-stage of pipeline CPU Design
- 3、Pipelined CPU with stall
- 4、Pipelined CPU with forwarding
- 5、Pipelined CPU resolving control hazard and support execution 31 MIPS Instructions

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures
- Precaution

Experiment Purpose 1

- Understand 31 MIPS instructions.
- Extend your design to support all the 31 MIPS instructions in pipelined CPU.

Experiment Purpose 2

- Understand why and when Control Hazards arise
- Master the methods of resolving Control Hazards
  - Freeze or flush
  - Predict-not-taken
  - Predict-taken
  - Delayed-branch
- Implement the predict-not-taken scheme in your final pipelined CPU.
- Write the verification program yourself to test whether it can execute right in both taken and not-taken cases and to test all the 31 instructions.

Pipelined CPU supporting execution of 31 MIPS instructions

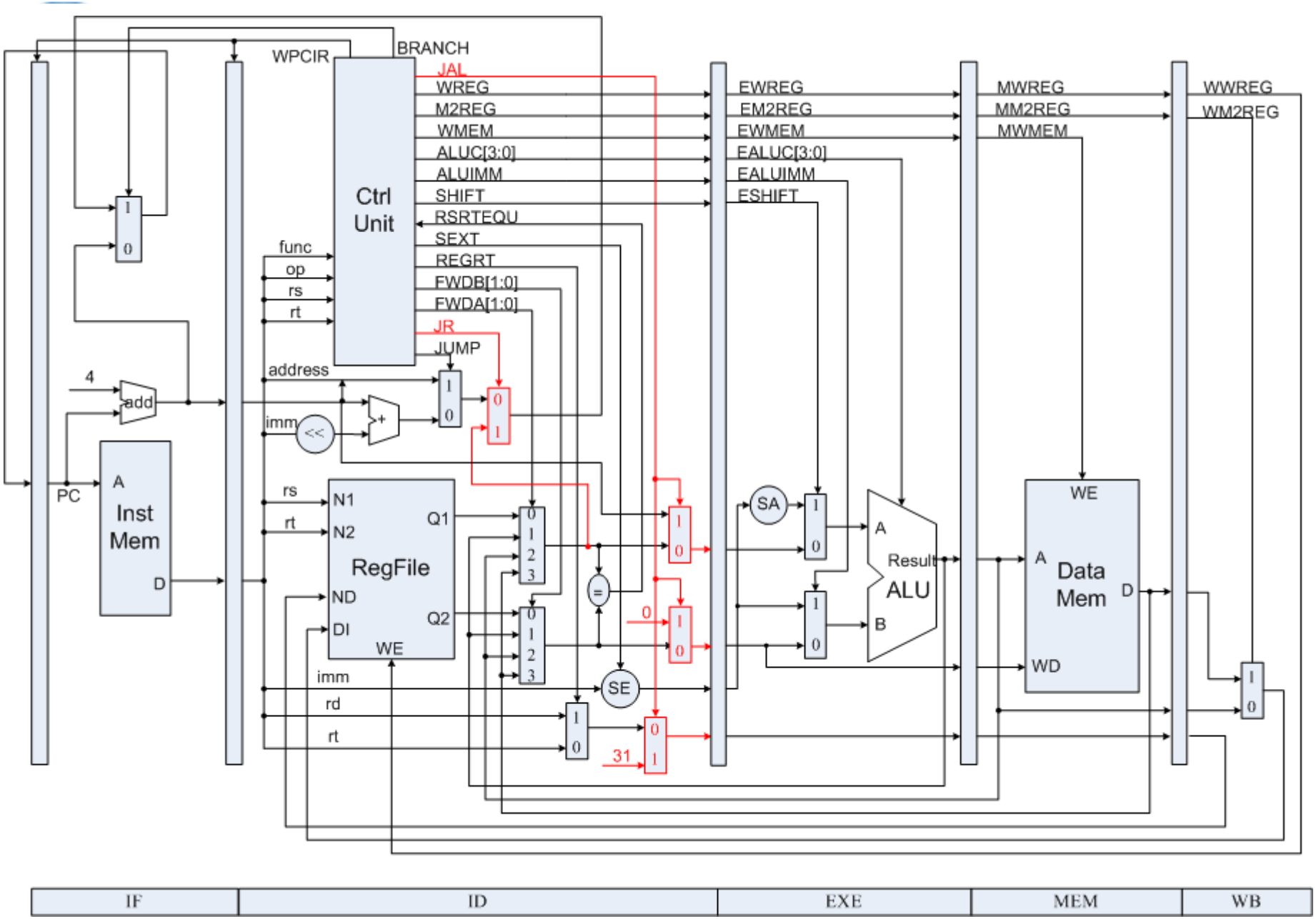
ZheJiang University

MIPS Instructions								
Bit #	**31..26**	**25..21**	**20..16**	**15..11**	**10..6**	**5..0**	**Operations**	
R-type	op	rs	rt	rd	sa	func		
add	000000	rs	rt	rd	00000	100000	rd = rs + rt; with overflow	PC+=4
addu		rs	rt	rd	00000	100001	rd = rs + rt; without overflow	PC+=4
sub		rs	rt	rd	00000	100010	rd = rs - rt; with overflow	PC+=4
subu		rs	rt	rd	00000	100011	rd = rs - rt; without overflow	PC+=4
and		rs	rt	rd	00000	100100	rd = rs & rt;	PC+=4
or		rs	rt	rd	00000	100101	rd = rs | rt;	PC+=4
xor		rs	rt	rd	00000	100110	rd = rs ^ rt;	PC+=4
nor		rs	rt	rd	00000	100111	rd = ~(rs | rt);	PC+=4
slt		rs	rt	rd	00000	101010	if(rs < rt)rd = 1; else rd = 0; <(signed)	PC+=4
sltu		rs	rt	rd	00000	101011	if(rs < rt)rd = 1; else rd = 0; <(unsigned)	PC+=4
sll		00000	rt	rd	sa	000000	rd = rt << sa;	PC+=4
srl		00000	rt	rd	sa	000010	rd = rt >> sa (logical);	PC+=4
sra		00000	rt	rd	sa	000011	rd = rt >> sa (arithmetic);	PC+=4
sllv		rs	rt	rd	00000	000100	rd = rt << rs;	PC+=4
srlv		rs	rt	rd	00000	000110	rd = rt >> rs (logical);	PC+=4
srav		rs	rt	rd	00000	000111	rd = rt >> rs(arithmetic);	PC+=4
jr		rs	00000	00000	00000	001000		PC=rs

Pipelined CPU supporting execution of 31 MIPS instructions

ZheJiang University

MIPS Instructions								
Bit #	**31..26**	**25..21**	**20..16**	**15..11**	**10..6**	**5..0**	**Operations**	
I-type	op	rs	rt	immediate				
addi	001000	rs	rt	imm			rt = rs + (sign_extend)imm; with overflow	PC+=4
addiu	001001	rs	rt	imm			rt = rs + (sign_extend)imm;without overflow	PC+=4
andi	001100	rs	rt	imm			rt = rs & (zero_extend)imm;	PC+=4
ori	001101	rs	rt	imm			rt = rs | (zero_extend)imm;	PC+=4
xori	001110	rs	rt	imm			rt = rs ^ (zero_extend)imm;	PC+=4
lui	001111	00000	rt	imm			rt = imm << 16;	PC+=4
lw	100011	rs	rt	imm			rt = memory[rs + (sign_extend)imm];	PC+=4
sw	101011	rs	rt	imm			memory[rs + (sign_extend)imm] <-- rt;	PC+=4
beq	000100	rs	rt	imm			if (rs == rt) PC+=4 + (sign_extend)imm <<2;	PC+=4
bne	000101	rs	rt	imm			if (rs != rt) PC+=4 + (sign_extend)imm <<2;	PC+=4
slti	001010	rs	rt	imm			if (rs < (sign_extend)imm) rt =1 else rt = 0; less than signed	PC+=4
sltiu	001011	rs	rt	imm			if (rs < (zero_extend)imm) rt =1 else rt = 0; less than unsigned	PC+=4
J-type	op	address						
j	000010	address					PC = (PC+4)[31..28],address<<2	
jal	000011	address					PC = (PC+4)[31..28],address<<2 ; $31 = PC+4	


WPCIR
BRANCH
JAL
WREG
M2REG
WMEM
ALUC[3:0]
ALUIMM
SHIFT
RSRTEQU
SEXT
REGRT
FWDB[1:0]
FWDA[1:0]
JR
JUMP
Ctrl Unit
func
op
rs
rt
address
imm
4
add
PC
A
Inst Mem
D
N1
N2
Q1
RegFile
ND
DI
Q2
WE
rs
rt
imm
rd
rt
31
SE
EWREG
EM2REG
EWMEM
EALUC[3:0]
EALUIMM
ESHIFT
SA
A
Result
ALU
B
MWREG
MM2REG
MWMEM
WE
A
Data Mem
D
WD
WWREG
WM2REG
IF
ID
EXE
MEM
WB

Careful with JAL and LUI

- Need add data path for JAL and LUI
- JAL:
  - How many stall ?
  - When write R31? What write to R31 ?

Observation Info

- Input
  - West Button: Step execute
  - South Button: Reset
  - North Button + 4 Slide Button: Register Index
  - East Button: High 16-bit of Reg. Content Sel.
- Output
  - 0-7 Character of First line: Instruction Code
  - 8 of First line : Space
  - 9-10 of First line : Clock Count
  - 11 of First line : Space
  - 12-15 of First line : Register Content
  - Second line : “stage name”/ number / type: stage name: 1-”f”, 2-”d”, 3-”e”, 4-”m”, 5-”w”(high bits is ignored)

Precautions

- 1、According to datapath to modify the modules of top module, then modify the definitions of the modules, then modify the CPU Controller.
- 2、The signals which transfer from former stage to later stage should be stored to stage registers temporally.
- 3、Design the program for verification by yourself, includes:
  - Stall
  - Forwarding
  - All kinds of instructions

Signed and unsigned type

- Reg and wire are unsigned type
- Signed type:
  - reg signed
  - wire signed
- Type conversion
  - $signed()
  - $unsiged()

Submission

- Lab report 5 named as StID_name_Lab5.doc (or pdf)

- All your verilog code, coe file, testing file, etc., in your project directory compacted as StID_name_prj.rar (from Lab 6)

-

- Thanks!