ENGR 3410: Midterm

Mark L. Chang

November 7, 2005

Instructions

This is a test. This is only a test.

You have 3 hours to complete this exam. You should take this exam in one sitting. You should not need anywhere near 3 hours. You may use any sources for solving your problems except for other people. This includes the internet, Google^{TM}, a calculator, lecture notes, and perhaps most importantly, the textbook.

The exam is due at 5PM, Friday, November 11th, 2005. You may slide the exam under my door in East Hall, or hand it to me directly. Partial credit is given, so please show all work. Be clear with any assumptions you make.

Problem 0

Take the following bit pattern:

```
1010 1100 1010 1000 1111 1111 1111 1101
```

what does it represent, if we assume that the number is:

- a two's complement integer?
- an unsigned integer?
- a single precision floating point number?
- a MIPS instruction?

You might find Appendix A handy, especially page A-50 and beyond. You will find the appendix in the following folder on the Olin network:

```
\\fsvs01\StuFac\CompArch\Book CD\Content\COD3e\CDSections
```

Problem 1

Bono (of U2 fame) wrote a program to model microfluidic flow that takes 100 seconds to run on the 1GHz machine in my office. Of course, The Edge (also of U2 fame) tweaks the compiler to optimize Bono's program by replacing all instances of multiplying a value by 4 (ie. $n * 4$) with two sequential adds ($m = n + n$, $p = m + m$).

On this machine, the CPI of a multiply instruction is 5, and the CPI of an add instruction is 1. After The Edge recompiles the program with the optimization, it runs in 85 seconds on the same machine.

Impressed, Bono tries to stump The Edge by asking him this question: *How many multiplies were replaced by the new compiler?* Help The Edge out by showing an answer to Bono's question.

Problem 2

The instruction `swap $rs, $rt` can be implemented with the following three MIPS instructions:

```
addi $rd, $rs, 0
addi $rs, $rt, 0
addi $rt, $rd, 0
```

Axl Rose, among his many musical talents, is a fan of hardware design. He tells you that he can add hardware support for the `swap` instruction to our single-cycle CPU, but that it would increase the clock period by 15%. For a given program, what must the percentage of swap instructions be in order to recommend using Axl's modified CPU design?

Problem 3

Natalie Merchant, the former 10k Maniac, wants to break with RISC tradition and implement an arithmetic instruction that can directly access memory. Specifically, she wants the following instruction:

```
addm $1, 100($2)  # $1 = $1 + Memory[$2 + 100]
                  # addm rt, imm16(rs)
```

What modifications to our *single-cycle* processor (if any) are necessary to support Natalie's *addm* instruction? Show the RTL, any data path changes, and settings for control signals. If it is not possible to implement, explain why. Single cycle worksheet is at the end of this exam.

Problem 4

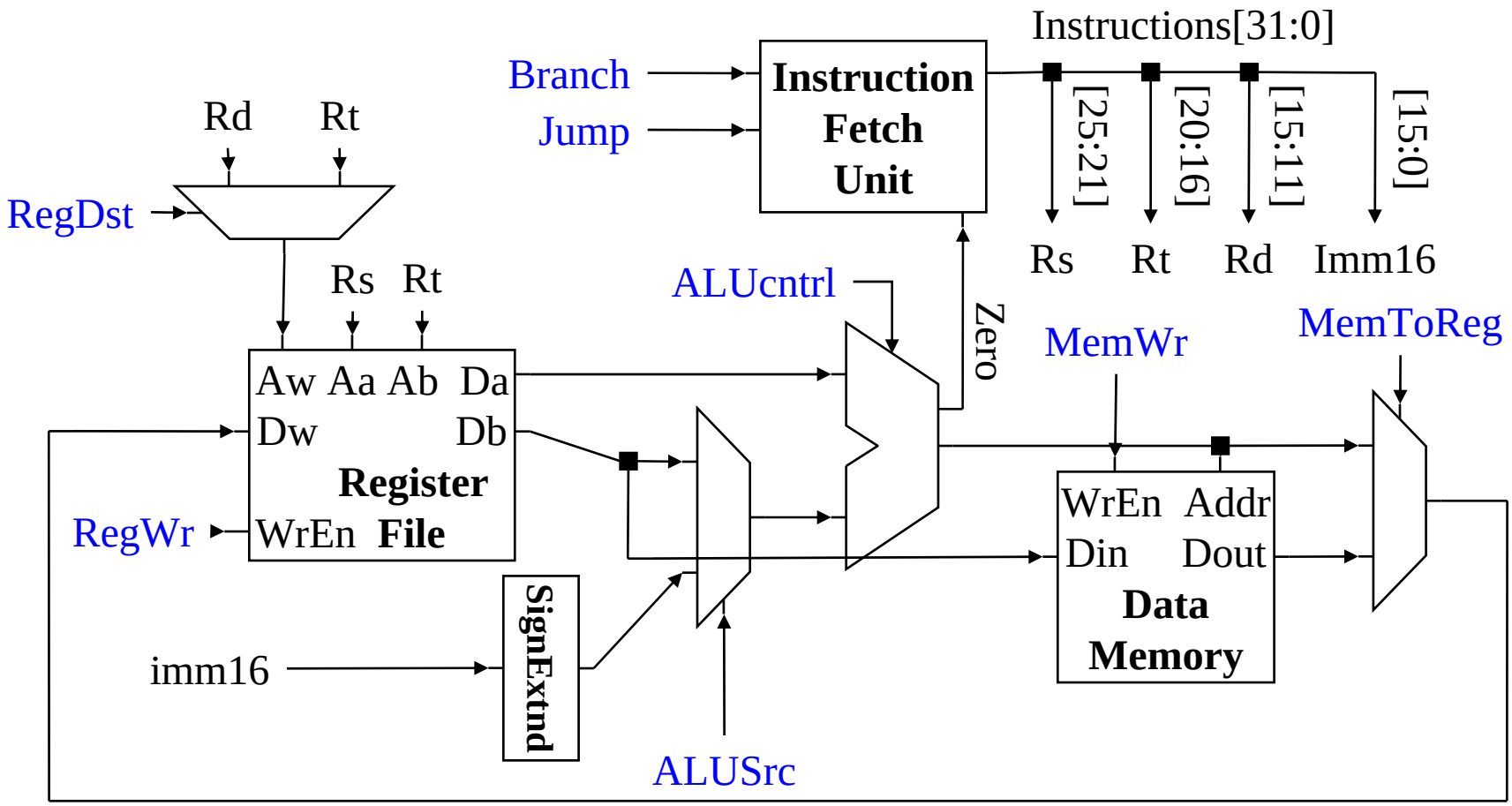
Not be left out of the party, Sheryl Crow, the former school teacher, wants our multi-cycle CPU to be modified to support the *addm* instruction. Show the RTL, any data path changes, and control signal modifications. You can use the multicycle data path and control worksheets at the end of the exam.

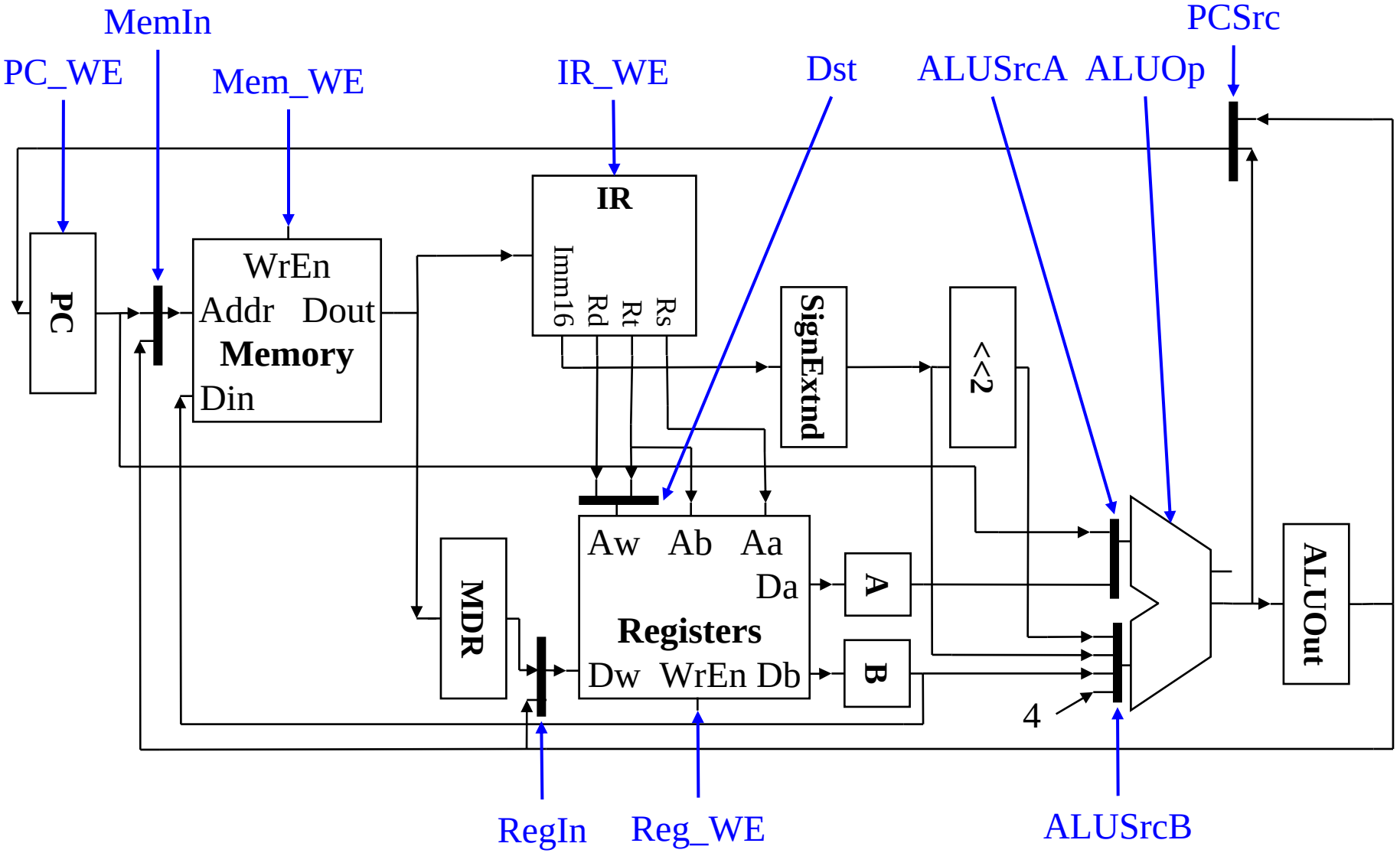
```
addm $1, 100($2)  # $1 = $1 + Memory[$2 + 100]
                  # addm rt, imm16(rs)
```

Problem 5

Scott Weiland, formerly of Stone Temple Pilots and currently with the surprisingly good Velvet Revolver, has a wacky idea. Instead of doing a load word with the source address coming from a register, he wants to implement a new instruction, *ldni*, load next indirect. This instruction performs a load from memory, however, the address to load from is stored in memory (not a register) immediately following this instruction. Think of it as a "double wide" instruction.

Scott wants to know what changes you need to make to the multi-cycle CPU we did in class, both in terms of data path and control, as necessary. He'd also love to see some RTL.


Instructions[31:0]
Branch
Jump
Instruction Fetch Unit
[25:21]
[20:16]
[15:11]
[15:0]
Rs
Rt
Rd
Imm16
Rd
Rt
RegDst
Rs
Rt
ALUcntrl
Zero
MemWr
MemToReg
Aw Aa Ab Da
Dw Db
Register
RegWr
WrEn File
WrEn Addr
Din Dout
Data Memory
imm16
SignExtnd
ALUSrc


MemIn
PCSrc
PC_WE
Mem_WE
IR_WE
Dst
ALUSrcA
ALUOp
IR
Imm16
Rd
Rt
Rs
PC
WrEn
Addr
Dout
Memory
Din
SignExtnd
<<2
Aw
Ab
Aa
Da
Registers
Dw
WrEn
Db
A
B
MDR
ALUOut
4
RegIn
Reg_WE
ALUSrcB

	WE				ALU						
State	PC	Mem	Reg	IR	SrcA	SrcB	Op	Dest	MemIn	RegIn	PCSrc
1											
2											
3Br											
3Rt											
4Rt											
3St											
4St											
3Lo											
4Lo											
5Lo											
					A	<<2		Rt[20:16]	PC	MDR	ALU
					PC	SE		Rd[15:11]	ALUout	ALUout	ALUout
						B					
						4					