Name: Zack Smaridge

Student ID: 8517 1587

EEL 4713: Computer Architecture - Spring 2012
Midterm 1 - 100 points possible

8

1. (12 pts) What are three drawbacks for taking a single cycle to execute each instruction?

1) Shorter instructions still have to wait even though their done quicker
2) Single cycle instructions cannot be pipelined for greater speed, covered here
3) The critical path is much longer and results in a slower clock speed why is a slow clock bad

6

2. (6 pts) If you were to add a no-op instruction (the instruction performs no operation) to the MIPS ISA and considering the multi-cycle datapath, how many cycles would this instruction require?

2 cycles - Fetch and decode
Really it just needs enough time to add 4 to the current PC and then load it back in.

4

3. (5 pts) Which of the following statements are potential arguments for MIPS to have 32 general-purpose registers (GPRs) $r0-$r31 instead of 64 GPRs? Mark all that apply.

1) The ALU can be made smaller with 32 GPRs less hardware
2) The register file can be faster with 32 GPRs smaller muxes means less propagation delay
3) The use of a 32-bit architecture implies the use of 32 GPRs simpler but does not necessarily imply
4) Encoding of R-type instructions with 64 GPRs would lead to very few opcodes available in the 32-bit instruction word register addr would be larger leaving fewer bits for opcodes
5) The control logic is simpler with 32 GPRs should not affect logic larger bus to mux: 5 vs. 6 bits

6

4. (10 pts) The MIPS "jump and link" instruction implicitly stores the PC+4 value to a fixed register (R31). Discuss the implications, if any, if the ISA was changed such that any arbitrary register could be specified.

If an arbitrary register were chosen it would make things more complicated because we would have to remember which register we stored it in. This causes problems for the jump register operation which assumes the return address is in R31.

this inst. can jump to the value in any register

and jump distance is reduced

5. (7 pts) Which method, callee or caller saved registers, may result in fewer assembly instructions and why.

Caller saved registers because when there is a function call there will be fewer registers that need to be pushed and popped on the stack. Each register that needs to be kept means an extra 2 instructions (sw before, lw after).

0

6. (10 pts) In MIPS ISA the source registers are fixed (*$rs* and *$rt*), while the destination register can be *$rd* (R-type) or *$rt* (I-type). Why is this approach preferable to keeping the destination register fixed and the source registers different?

Did the way it was written

If the source registers were not fixed it would require more hard wire for the extra mux and we would not be able to speculatively load the registers. The processor would take longer because the registers would not be loaded until the instruction decode was complete.

10

7. (50 pts total) Consider the addition of a new instruction into the MIPS instruction set given the multi-cycle instruction execution implementation. This instruction, *inc2 $rd, $rt*, is used to automatically increment the values stored in both registers *$rd* and *$rt* in a single instruction (i.e., *$rd* = *$rd* + 1 and *$rt* = *$rt* + 1).

47

a) (10 pts) Discuss why this instruction cannot be supported by the current multi-cycle datapath?

7

This instruction cannot work because ① register rd cannot be accessed via a read (only a write) ② the ALU does not have an increment function nor is there a way to load a 1 into the ALU.

It is not necessary but would make the execution time faster if there were an additional register between the register file and ALU to temporarily store the operand (allows for pipelining)

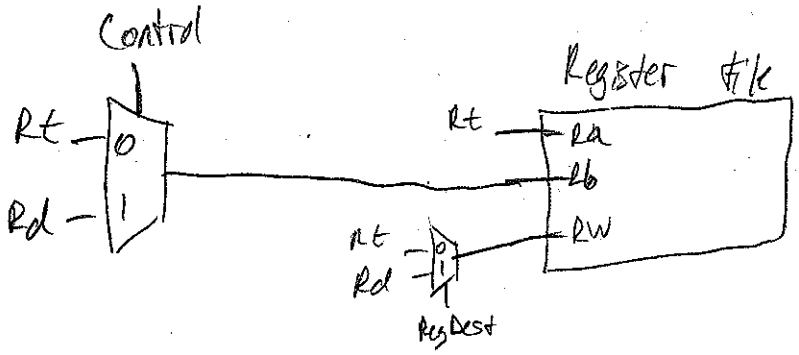
no mention of 2 ALU ops + 2 reg. file writes in 1 inst.

81/100

b) (15 pts) If the instruction cannot be supported by the current multi-cycle datapath, discuss the architectural and control logic changes, at a high level, that are necessary to support this instruction.

15

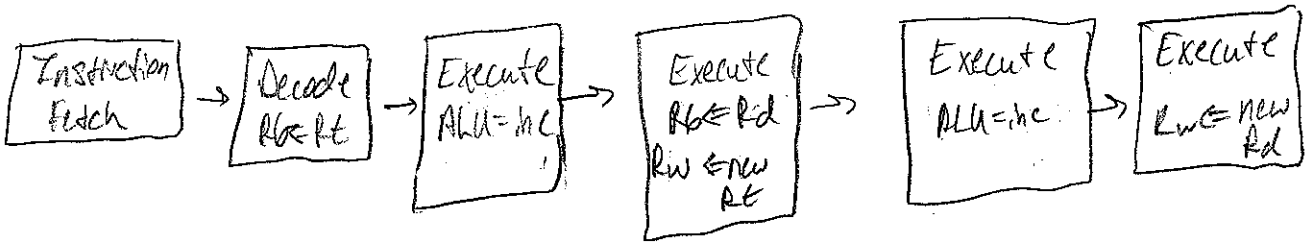
① Add a 2-to-1 Mux going into the Register File where Rs is currently. Add a 1-bit control signal to specify which bit the Mux loads, Rd or Rs.



② Add an ALU function that increments ib by 1. This is a pretty straight forward step in VHDL.

when opcode = increment
output ⇐ ib + 1;

③ The state machine will need an extra 2 states to allow for both registers to be loaded and incremented. Should look like this:



d) (15 pts) How many cycles will this new instruction require? What will be done on each cycle?

15

It should take five cycles.

(1) Instruction fetch
(2) Decode - new mux speculatively loads Rt onto busB
(3) Execute - ALU selects increment and Target register accepts value of rt+1
(4) Execute/Store - new mux loads Rd onto busB while Rw selects Rt register to write the new value in from the target register
(5) Execute - ALU selects increment and Target register accepts value of rd+1
(6) Store - Rw selects Rd register to write the new value in from the Target register

e) (5 pts) Does the clock cycle time need to change to support this new instruction?

5

The clock speed should be fine as the new instruction is using the ALU the same as the others

f) (5 pts) What instruction type (I-, R-, J-type) would be most appropriate for this instruction and why?

5

This should be an R-type instruction because it uses the Rd register which is only accessible via the R-type format. Also makes sense to use the R-type so that the ALU function may be specified in the "funct" field rather than taking up more "opcode" space.