CPSC 3300 – Exam 2 Sample Questions

1. Matching. Write the correct term from the list into each blank. (2 pts. each)

control signal structural hazard control hazard speculative execution

out-of-order execution load-use data hazard forwarding IPC (instruction-per-cycle)

(a) \_structural hazard\_\_\_\_\_\_\_ when hardware cannot support the combination of instructions we want to

execute in the same clock cycle

(b) \_control signal\_\_\_\_\_\_\_\_\_\_ value used for selecting a mux input or selecting the operation of a functional

unit

(c) \_forwarding\_\_\_\_\_\_\_\_\_\_\_\_ providing a data value to any unit where it is needed after the data value has

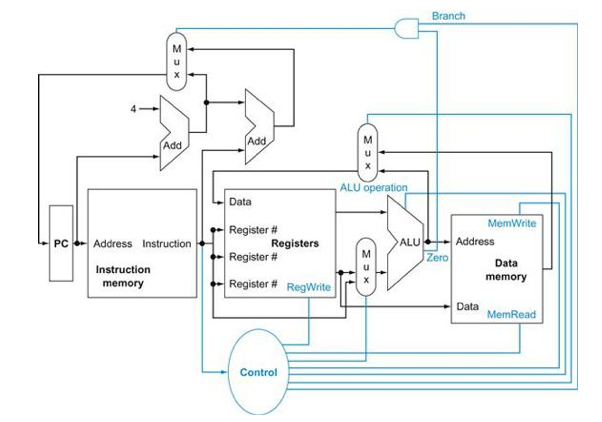
been produced but before it is available in the register file

(d) \_speculative execution\_\_\_\_ allowing an instruction that is control dependent on a branch to execute after

the branch direction is predicted and before the branch is resolved

(e) \_IPC (instruction-per-cycle)\_ the measure of performance of (advanced) processor pipeline

(f) \_out-of-order execution\_\_\_\_ allowing instructions behind a stalled instruction to proceed to execution



❸❸

❷

❶

2. Consider the MIPS “and” instruction as implemented on the datapath above (Figure 4.2 from textbook):

and R1, R2, R3 // Reg[1] <- Reg[2] & Reg[3]

Circle the correct value 0 or 1 for the control signals (a-d) and circle whether each of the three muxes (e-g)

selects its upper input, lower input, or don't care. For the ALU operation (h) circle one of the function names.

(The Zero condition signal will be assumed to be 0.) (8 pts.)

(a) Branch = 0 1 (e) Mux1 (upper left; output to PC) = upper, lower, don't care

(b) MemRead = 0 1 (f) Mux2 (upper middle; output to Data port of Regs) = upper, lower, don't care

(c) MemWrite = 0 1 (g) Mux3 (lower middle; output to bottom leg of ALU) = upper, lower, don't care

(d) RegWrite = 0 1 (h) ALU operation = and, or, add, subtract, set-on-less-than, nor

3. Identify the five stages of the simple pipeline we studied, and explain what each stage does when

processing the and instruction from question 2 above. (9 pts.)

and R1, R2, R3 // Reg[1] <-- Reg[2] & Reg[3]

Instruction fetch: load instruction from instruction memory, increment program counter by 4

Decode: decode instruction, read registers R2 and R3 for operand values

Execute: ALU performs AND operation using operands, generates result

Memory: No memory access, result bypasses data memory unit

Writeback: Result is written to register R1

4. Associate each term or statement below with a type of dependency. Circle one or more of RAW, WAR, or WAW. (Destination registers are listed first for add and subtract instructions.) (3 pts. each)

(a) RAW / WAR / WAW true data dependency

(b) RAW / WAR / WAW false data dependency (name dependency)

(c) RAW / WAR / WAW add R3,R1,R2 followed by sub R1,R3,R4

(d) RAW / WAR / WAW add R3,R1,R2 followed by sub R5,R3,R4

(e) RAW / WAR / WAW type of dependency that can cause a load-use penalty

5. Provide short answers for the following questions. (2 pts. each)

1. Does internal forwarding always eliminate all stall cycles due to data hazards in a pipeline? You can justify/explain your answer by using a simple assembly or pseudo code.

No: Load-use data hazards lw r1, 0(r0)

add r3, r2, r1

1. What does the Branch Target Buffer (BTB) store, and how is this information used?

The Branch Target Buffer stores the locations of branch instructions and the respective branch target address for those branches. When a particular branch is predicted to be taken the Program Counter is updated with the branch target address from the BTB for that particular branch instruction during the instruction fetch causing the next instruction to be fetched to be the first “branch taken” instruction.

1. Given a simple 5-stage MIPS pipeline with single issue, what is the ideal IPC and for which scenario?

1 IPC, when no stalls occur

1. What hazard does it resolve by using an instruction memory and a data memory in the datapath?

Structural hazard

1. When the MIPS pipeline uses extra hardware to compare two register values in the ID stage instead of using the ALU in the EXE stage, what hazard does it address and what does improve?

Control hazard, reduces branch delay / misprediction penalty

(f) Consider a two-bit history for branch prediction. It records the state of the last branch as taken (T) or untaken (U) and predicts the next branch. Assume the two-bit is initialized to Weakly Not Taken (01). Determine the prediction on the following branch trace.

Actual : T T T T U T T T T U

Prediction: U T T T T T T T T T

6. Draw the dependency diagram for the following MIPS code. Destination registers are listed first except for the

sw (store word) instructions; sw writes into memory rather than a register. (8 pts.)

$6

$6

i1: loop: lw $1, 0($6)

i2: lw $2, 4($6)

i3: beq $1, $2, 3

i2: lw

i1: lw

i4: addi $6, $6, 4

$6

i5: bne $6, zero, loop

$2/RAW

$1/RAW

i4: addi

i3: beq

$6/RAW

i5: bne

7. The multi-cycle and pipelined datapaths that we have discussed in class have generally been broken down into 5 steps:

1. Hardware to support an instruction fetch
2. Hardware to support an instruction decode (i.e., a register file read)
3. Hardware to support instruction execution (i.e., the ALU)
4. Hardware to support a memory load or store
5. Hardware to support the write back of the ALU operation back to the register file

Assume that each of the above steps takes the amount of time specified in that table below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fetch | Decode | Execute | Memory | Write back |
| 305 ps | 275 ps | 280 ps | 305 ps | 250 ps |

1. Given the times for the datapath stages listed above, what would the clock period be for the pipelined datapath? [4pts]

305 ps

1. What is the maximum possible speedup of a pipelined implementation when compared to a single-cycle datapath implementation without the pipeline? [4pts]

1415 ps / 305 ps = ~4.639

8. Give the pipeline cycle diagram for the code segment given in question 6 above for the 5-stage pipeline with forwarding. Use arrows to mark the necessary forwarding. (10 pts.)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| loop: lw $1, 0($6) | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
| lw $2, 4($6) |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |
| beq $1, $2, 2 |  |  | F | - | - | D | E | M | W |  |  |  |  |  |  |
| addi $6, $6, $4 |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |
| bne $6, zero, loop |  |  |  |  |  |  | F | - | D | E | M | W |  |  |  |

9. The branch CPI penalty is calculated as extra CPI = (branch freq.)\*(misprediction freq.)\*(mispredict penalty)

(a) Which one of the three terms in the penalty equation will techniques like loop unrolling and predication reduce?

(3 pts.)

Branch frequency

(b) Which one of the three terms in the CPI formula does dynamic branch prediction attempt to reduce? (3 pts.)

Misprediction frequency

1. 2-issue processors. You are given the following code for a loop:

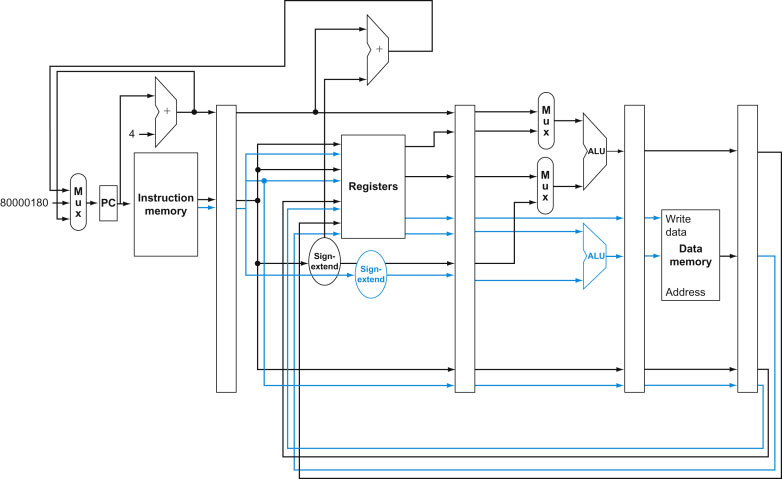
loop: lw t0, 0(s1)

addu t0, t0, s2

sw t0, 0(s1)

addi s1, s1, -4

bne s1, zero, loop



1. [4pt]If the loop exits after executing only two iterations. Draw a pipeline diagram for the given MIPS code on a 2-issue processor shown above. Each issue packet may contain up to one ALU/branch instruction and up to one load/store instruction. Assume the processor has perfect branch prediction.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| lw t0, 0(s1) | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
| addu t0, t0, s2 |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |
| sw t0, 0(s1) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |
| addi s1, s1, -4 |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |
| bne s1,zero,loop |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |
| lw t0, 0(s1) |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |
| addu t0, t0, s2 |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |
| sw t0, 0(s1) |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |
| addi s1, s1, -4 |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |
| bne s1,zero,loop |  |  |  |  |  |  |  |  |  |  | F | D | E | M | W |

1. [4pts]Rearrange the code to achieve better performance on a 2-issue statically scheduled processor from Figure above. Write the sequence below.

lw t0, 0(s1)

addi s1, s1, -4

addu t0, t0, s2

sw t0, 4(s1)

bne s1, zero, loop

1. [4pts]Repeat step a for the code from b.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| lw t0, 0(s1) | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
| addi s1, s1, -4 | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
| addu t0, t0, s2 |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |
| sw t0, 4(s1) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |
| bne s1,zero,loop |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |
| lw t0, 0(s1) |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |
| addi s1, s1, -4 |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |
| addu t0, t0, s2 |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |
| sw t0, 4(s1) |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |
| bne s1,zero,loop |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |