

UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Computer Architecture
ECE 568/668

Sample Midterm I

Israel Koren
Fall 2009

ECE568/Koren Sample Midterm.1 .1

Copyright 2009 Koren UMass

1. The cost of a pipeline can be estimated by $c+k*h$ where c is the cost of all logic stages, h is the cost of the latches in a single stage and k is the number of stages.

A pipeline performance/cost ratio (PCR) has been defined as $PCR = \text{Throughput}/(c+kh)$ where $\text{Throughput}=1/t$ and $t = t_{\text{latch}}+T/k$. T is the total time required for the nonpipelined execution. Derive an expression for the optimal number of pipeline stages, k_{opt} , that maximizes the PCR.

ECE568/Koren Sample Midterm.1 .2

Copyright 2009 Koren UMass

2. Suppose 25% of all instructions in a program are conditional branches (of which 65% are taken), and 10% are unconditional jumps and calls. Consider a four-deep pipeline, with each stage of the pipeline being executed in one cycle. Thus, each instruction takes four cycles in all to be executed, in the first of which the instruction is fetched. The branch is resolved at the end of the second cycle for unconditional jumps and calls, and at the end of the third cycle for conditional branches. Assuming that only the first pipeline stage can always be done independently of whether the branch is taken, and ignoring all other pipeline stalls, how much faster would the machine be without any branches in the workload? Also assume that without any branches, the machine completes on the average, one instruction every 1.3 clock cycles.

3. A given pipelined processor executes floating-point instructions of the type $F_i \leftarrow F_j \text{ op } F_k$. It has an I stage (for instruction fetch) and a D stage (for instruction decode and operand preparation) each taking one clock cycle. The execution phase of a floating-point ADD operation and a floating-point Multiply operation takes 2 and 4 cycles, respectively. Each execution is followed by a single (clock) cycle write back (W) into the floating-point register file which can support simultaneous reads but only a single write per cycle. The processor executes the following sequence of floating-point instructions:

| | |
|--------------------------------|--------------------------------|
| S1: $F_2 \leftarrow F_1 + F_4$ | S4: $F_4 \leftarrow F_1 * F_2$ |
| S2: $F_3 \leftarrow F_2 * F_4$ | S5: $F_1 \leftarrow F_2 + F_3$ |
| S3: $F_3 \leftarrow F_4 + F_5$ | S6: $F_4 \leftarrow F_1 * F_2$ |

(a) Assume that the processor has a single floating-point adder and a single floating-point multiplier. Show all data and structural hazards that may occur while executing the given program. Indicate the exact type of each hazard.

S1: $F2 \leftarrow F1 + F4$ S4: $F4 \leftarrow F1 * F2$
 S2: $F3 \leftarrow F2 * F4$ S5: $F1 \leftarrow F2 + F3$
 S3: $F3 \leftarrow F4 + F5$ S6: $F4 \leftarrow F1 * F2$

(c) Assume that the floating-point adder (with execution time of 2 cycles) and the floating-point multiplier (with execution time of 4 cycles) can operate in parallel and are fully pipelined (with a throughput of 1). Show the exact timing for the above program on the charts below for the following alternatives:

(I) Basic pipeline with data forwarding. (II) Software rescheduling and then executing on a pipeline with data forwarding.

(III) Dynamic scheduling in hardware using Tomasulo's algorithm with two reservation stations for each execution unit.

Make and justify any additional assumptions that are needed to your opinion.

(I)

| | (I) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | |
|-----------------------------|----------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
| S1: $F2 \leftarrow F1 + F4$ | S ₁ | I | D | + | + | W | | | | | | | | | | | | | | | | | | |
| S2: $F3 \leftarrow F2 * F4$ | S ₂ | | | | | | | | | | | | | | | | | | | | | | | |
| S3: $F3 \leftarrow F4 + F5$ | S ₃ | | | | | | | | | | | | | | | | | | | | | | | |
| S4: $F4 \leftarrow F1 * F2$ | S ₄ | | | | | | | | | | | | | | | | | | | | | | | |
| S5: $F1 \leftarrow F2 + F3$ | S ₅ | | | | | | | | | | | | | | | | | | | | | | | |
| S6: $F4 \leftarrow F1 * F2$ | S ₆ | | | | | | | | | | | | | | | | | | | | | | | |

(II) Software rescheduling and then executing on a pipeline with data forwarding.

| (I) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | |
|---|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
| S1: $F2 \leftarrow F1 + F4$ | S_1 | I | D | + | + | W | | | | | | | | | | | | | | | | | |
| S2: $F3 \leftarrow F2 * F4$ | S_2 | | | | | | | | | | | | | | | | | | | | | | |
| S3: $F3 \leftarrow F4 + F5$ | S_3 | | | | | | | | | | | | | | | | | | | | | | |
| S4: $F4 \leftarrow F1 * F2$ | S_4 | | | | | | | | | | | | | | | | | | | | | | |
| S5: $F1 \leftarrow F2 + F3$ | S_5 | | | | | | | | | | | | | | | | | | | | | | |
| S6: $F4 \leftarrow F1 * F2$ | S_6 | | | | | | | | | | | | | | | | | | | | | | |

ECE568/Koren Sample Midterm.1.7

Copyright 2009 Koren UMass

(III) Dynamic scheduling in hardware using Tomasulo's algorithm with two reservation stations for each execution unit.

| (I) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | |
|---|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
| S1: $F2 \leftarrow F1 + F4$ | S_1 | I | D | + | + | W | | | | | | | | | | | | | | | | | |
| S2: $F3 \leftarrow F2 * F4$ | S_2 | | | | | | | | | | | | | | | | | | | | | | |
| S3: $F3 \leftarrow F4 + F5$ | S_3 | | | | | | | | | | | | | | | | | | | | | | |
| S4: $F4 \leftarrow F1 * F2$ | S_4 | | | | | | | | | | | | | | | | | | | | | | |
| S5: $F1 \leftarrow F2 + F3$ | S_5 | | | | | | | | | | | | | | | | | | | | | | |
| S6: $F4 \leftarrow F1 * F2$ | S_6 | | | | | | | | | | | | | | | | | | | | | | |

ECE568/Koren Sample Midterm.1.8

Copyright 2009 Koren UMass