Midterm Histogram

Average: 15.9

3/4 => 13
1/2 => 17
1/4 => 19

0.3
0 = HALT //

3 = HALT //

7 = FLIP-GUARD //

13 = LOOP-Top = — 13 //

Assume that the code starts at memory location 000. In the following code, fill in the blanks.

While the normal (decimal) version of that number, which are the values of certain labels, that is, what is the actual value that will be put into the machine instruction to designate that label.

Autumn 2020
Proto S. Hauck
Midterm Exam

EE 469

Name:
Consider the following two processors:

<table>
<thead>
<tr>
<th>Processor</th>
<th>CPI</th>
<th>Clock Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>1.5</td>
<td></td>
</tr>
</tbody>
</table>

If the processors have the same instruction set, which processor is the fastest and how much faster is it than the other processor? Please express as a ratio.

\[
\frac{1.5 \times CPI_2}{1.0 \times CPI_1 \times \text{clock period}} = \frac{15}{20} = \frac{3}{4}
\]

Thus, Processor P2 is \(\frac{3}{4}\) times faster than Processor P1.
(a) We create a 3-stage pipeline, with a first-stage IF, a second-stage EX, and third-stage WM. What is the impact on delay slots? 

(b) We break the EX stage into EX1 and EX2, with all math operations taking 2 cycles. What is the impact on throughput and delay slots? 

(c) What is the impact on performance? Assume we can pipeline without worrying about hazards, and have a 5-stage pipeline with a cycle time of 10ns. We switch from a 4-stage pipeline with a cycle time of 9ns, hence gain 40%. 

(d) In the previous scenario, assume we switched from a 2-stage pipeline with a cycle time of 10ns. What is the impact on performance?

(e) Assume we can pipeline without worrying about hazards, and have a 5-stage pipeline with a cycle time of 10ns. We switch to a 10-stage pipeline with a cycle time of 10ns. What is the cycle time of 10ns. The impact on throughput and delay slots? 

Delay of 50 ns; all instructions use the ALU.
ORR x0, x0, x2

LSL x2, x2, #40

LDR x0, [x1, #24]

LDR x1, [x3, #8]

LDUR x1, [x3, #0]

\[
\begin{array}{c|c|c|c|c|c}
0 & 1 & 2 & 3 & 4 & 5 \\
\hline
8 & B & 0 & 1 & 2 & 3 \\
\end{array}
\]

whenever registers you want, but your program should be as short and efficient as possible.

4) LDUR x0 [X31, #3] is an illegal instruction because the address is not aligned.
5.) Create a single-cycle processor that can do "idur" and "mul" only. Draw the datanpath for this machine (including the internal structure of the instruction fetch unit). Note that your machine should be as simple as possible - if there is ANYTHING you don't need, don't include it. Also, you can use a multiplier block as a unit, you don't have to build one yourself.

Instruction = MEM[PC]

Add = REG[ERw] + SEC[Add9]

PC = PC + 4

ADD = REG[RD] * REG[ERw]
Accelerator

Branches

Lecture

CBZ x10, -6

ADD x10, xo, x1

Accelerator

Branches
ILP = Instruction-Level Parallelism

Advanced processors require significant hazard avoidance/flexibility.

Key to pipelining was dealing with hazards.

Readings: 4.10

Instruction-Level Parallelism & Advanced Architectures
Instructions per cycle: IPC

$CPI = \frac{1}{4}$

What if we want a CPI < 1.0?
Reduce CPI

Greater pipelining means more hazards, delay slots
Reduce clock period by heavy pipelining

Advanced processors optimize two factors:

Why ILP
ILP Example

Source code:

1. ADDI x9, x0, #15
2. SUB x2, x3, x0
3. EORI x1, x4, x0
4. ORR x4, x3, x0

Constraint graph:

Redundant

Do these instructions place to happen in that order?