Computer Architecture and Programming Languages
Jacobs University Bremen
Fjolla Dedaj

Course: CO20-320241
2nd of December, 2019

Homework 11

Problem 11.1

Use the data from the example given on slide 29(Lecture 20 & 21) and assume that you have to execute a program containing 2 load instructions, 1 store instruction, 3 R-format instructions and 1 branching instruction for computing the speedup relate to the execution time in the following scenarios:

(a) multi-cycle approach compared to single cycle approach,
(b) pipelined approach compared to single cycle approach,
(c) pipelined approach compared to multi-cycle approach.

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load Word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store Word (sw)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (add, sub, and, or, slt)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (beq)	200 ps	100 ps	200 ps			500 ps

Solution:
Single Cycle Approach:
Find the instruction that takes the longest to complete and multiply that with the total number of instructions because each instruction will take the same amount of time. Thus, we have:

$$(2 + 1 + 3 + 1) \times 800 = 7 \times 800 = 5600$$

Multi-Cycle Approach:
Each instruction will take its respective amount of time to complete. Therefore, we have:

$$2 \times 800 + 1 \times 700 + 3 \times 600 + 1 \times 500 = 4600$$

Pipelined Approach:
The instructions happen concurrently, however, we have a delay for the instruction fetch:

Instructions				Total
Load: 800ps	Store: 700ps	R-format: 600ps	Branch: 500ps	2600ps
Bubble: 800ps	Load: 800ps	R-format: 600ps		2200ps
Bubble: 800ps	Bubble: 800ps	R-format: 600ps		2200ps

The processing stage taking the longest is: 2600ps

Calculate the time rates:

(a) $\frac{single-cycle}{multi-cycle} = \frac{5600}{4600} = 1.2173913$

(a) $\frac{single-cycle}{pipelined} = \frac{5600}{2600} = 2.15384615$

(a) $\frac{multi-cycle}{pipelined} = \frac{4600}{2600} = 1.76923077$

Problem 11.2

Solution:
(a) the string $zero:
$zero

(b) all strings which start with a and end with b and may contain any other letters or digits:
\a[a-zA-Z0-9]*\b

(c) all strings which start and end with a digit and may contain letters, digits, and underscores:
[0-9][a-zA-Z0-9_]*[0-9]

(d) all strings which contain only the characters a and b, start with abb following by at least 4 a, the total length of the string should not be longer than 10 characters:
"abb"a{4}[ab]{0,3}

(e) all positive integer numbers:
[1-9]+[0-9]*

(f) all integer numbers:
[-]?[1-9]+[0-9]*

(g) all positive floating point numbers:
[0-9]*[.]?[0-9]*

(h) the strings pit, spot, spate, slap two, respite but it should not recognize the strings pt, Pot, peat, part:
(**pit**|**spot**|**spate**|**slap two**|**respite**)

Problem 11.3

Solution:
Correct answers are denoted in bold.

(a) ab+c? (including ?)

(1) abc
(2) ac
(3) abbb
(4) bbc

(b) a.[bc]+

(1) abc
(2) abbbbbbbb
(3) azc
(4) abcbcbcbc
(5) ac
(6) asccbbbbcbcccc

(c) (very)+(happy)?((CS)—(IMS)—(ECE)) student

(1) very happy student
(2) happy CS student
(3) very very happy ECE student
(4) very very very happy IMS student
(5) very very very very IMS student

(d) < [∧ >] + >

(1) <**an xml tag**>
(2) <**opentag**> <**closetag**>
(3) <**/closetag**>
(4) <>
(5) <**with attribute** >