2GA3 Tutorial #8

DATE: November 12th, 2021

TA: Jatin Chowdhary

Quick Stuff

- Everyone's asking: ''When will you post tutorial #7 slides?''
  - Answer: In ~1 day (tomorrow)
- Tutorial #8 slides will (also) be posted tomorrow
  - Why the delay?
    - Busy week
    - LibreOffice being a pain
    - I decided to sleep in this morning
      - Thanks Liam!

Review Questions

- No time
  - I'll add questions (and answers) in later, and you can review them on your own
- Should I double the difficulty of review questions? Or leave them as is?
  - Thoughts?

Listen Carefully

Adapt & Improve

- **Tip Of The Day:** *Study smarter, not harder*
- Learn from midterm #1
  - What kind of questions were on midterm 1?
    - *Stuff from lecture, tutorial, assignment, etc.*
  - What content should you focus on?
    - *Theoretical questions or calculation questions?*
  - What was your biggest issue with midterm 1?
    - *Time? If so, congregate and organize your notes so they are easy to access and reference*
      - *Most of you probably wasted a lot of time going through your unorganized notes and Googling. BAD!*

Midterm #2

- Mostly on chapter 3 and 4
  - *Organize your notes based off these 2 chapters*
  - *First 2 questions are chapter 3; last 2 questions are chapter 4*
- Only 4 questions
  - **Tip:** Think about the questions you've done so far?
    - IEEE Floating Point
      - *We did lots of these*
    - Datapath
      - *I'll do another one today*
    - Pipelining
      - *This is all from tutorial #8 (today)*
    - *I've already given you 3 questions on the midterm! (Kind of)*
- Y'all better get 100%
  - (This is the expectation; rather, what I expect from my sections)

Questions?
Comments?
Concerns?

The Problem

- Recently, I realized that most of you are NOT on campus
  - Most students are commuting
    - I wish I had known sooner! But nobody tells me anything
    - I made an incorrect assumption – you can never underestimate how wrong you can be
- The problem is: Commuting is a huge time sink
  - Hence, nobody wants to do it – including me!
    - i.e. A 30 minute commute (one-way) results in 20+ hours wasted every month
      - Time >>> Money
  - And if I don't wanna commute, I can't expect you to do it either

Lesson Learned

- And if I don't wanna commute, I can't expect you to do it either – at least not in good faith
  - Translation: You are not going to do what I tell you to do, you are going to do what you see me do
    - *i.e. Some students are emailing me at 3-4 AM – which is great*
- The greatest thing I've learned, from you, is:
  - Your students/children/employees will NOT do what you tell them to do, rather, what they see you do
    - *i.e. The early morning emails*

The Solution

- I have a few ideas
  - Should've done them sooner
- The goal is:
  - Break up the content into *bits* that are easy to learn
    - Modularization
    - Everyone's got different questions
  - Maximize efficiency and minimize time
  - Create long lasting value for everyone
    - The answer/solution should be clear, unambiguous, and immediately obvious
- This sounds a bit ominous but you'll see – more to come…

Learning Time

(Straighten Up)

Quick Tip

- **Question:** How to tell if a number is odd or even in binary?
- **Answer:**
  - Look at the last 2 bits!
    - This is helpful when the question says, ‘‘round to the nearest even/odd/etc.’’
    - You won’t have to convert to decimal, and then back to binary
  - If the binary sequence ends in 00 or 10, then it is e___
    - Next slide!

Quick Tip

- Table of Binary & Decimal Numbers:

Binary	Decimal	Binary	Decimal
0000 0000	0 (special case)	0000 1001	9
0000 0001	1	0000 1010	10
0000 0010	2	0000 1011	11
0000 0011	3	0000 1100	12
0000 0100	4	0000 1101	13
0000 0101	5	0000 1110	14
0000 0110	6	0000 1111	15
0000 0111	7	0001 0000	16
0000 1000	8	0001 0001	17

Datapath

- I wanna quickly review a question from last week's tutorial
  - Why? Because it's probably gonna be on the midterm!
    - IEEE Floating Points
      - *We've done lot's of these*
    - **Datapath**
      - *I'll do one (quickly) right now*
    - Pipelining
      - *Today's tutorial, or what's left of it*
  - If you understand these 3 things, you'll easily get an 85+ on the midterm. The remaining 15% is on YOU!

Review Question (T7)

- **Question:** *Refer to datapath.png*
- **Answer:**
  - 4.5.0
    - Convert *0x00c6aa23* to binary
      - Answer: 11000110101010100**0100011**
        - Start at **rightmost bits**, get instruction *(i.e.* ***sw****)*, and then decode the rest
        - Use RISC-V instruction list to convert from opcode to instruction
  - Remember: *0x00c6aa23* will be stored "*backwards*" in memory, because RISC-V is little endian
    - *0x00000008* | *23 aa c6 00*

      Address | *Value*

Review Question (T7)

- **Question:** *Refer to datapath.png*
- **Answer:**
  - 4.5.1
    - *From lecture slides!*
  - What is ALUop?
  - What is ALU control?

ALU Control

- Assume 2-bit ALUOp derived from opcode
  - Combinational logic derives ALU control

opcode	ALUOp	Operation	Opcode field	ALU function	ALU control
ld	00	load register	XXXXXX	add	0010
sd	00	store register	XXXXXX	add	0010
beq	01	branch on equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001

Review Question (T7)

- **Question:** *Refer to datapath.png*
- **Answer:**
  - 4.5.2
    - The new program counter address is:
      - $PC_{new} = PC_{old} + 4$
        - *4 means 4 bytes*
    - Since we are not jumping around or anything (i.e. Going to a different function), we simply go to the next instruction
      - *No branching; so go to next instruction*

Review Question (T7)

- **Question:** *Refer to datapath.png*
- **Answer:**
  - 4.5.3
    - ALUsrc
      - Input: x12 (Register), 0x00000014 (Immediate value = 20)
      - Output: 0x00000014 (Immediate value = 20)
    - MemToReg
      - Input: x13 + 0x00000014 (Register + Offset = Memory Address)
      - Output: Homework
        - *Remember, we are not writing to a register*

Review Question (T7)

- **Question:** *Refer to datapath.png*
- **Answer:**
  - 4.5.4
    - *Homework*

Review Question (T7)

- **Question:** *Refer to datapath.png*
- **Answer:**
  - 4.5.5
    - Read register 1 = x13
      - Base address
    - Read register 2 = x12
      - The data we are trying to store
    - Write register, Write data, RegWrite
      - All of them are *X*, because we are not interested in writing to the register. The ***sw*** instruction should not write back
      - *Note: RegWrite is false*

Tutorial Question #1

- **Question:** In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
250 ps	350 ps	150 ps	300 ps	200 ps

- Also, assume that instructions executed by the processor are broken down as follows:

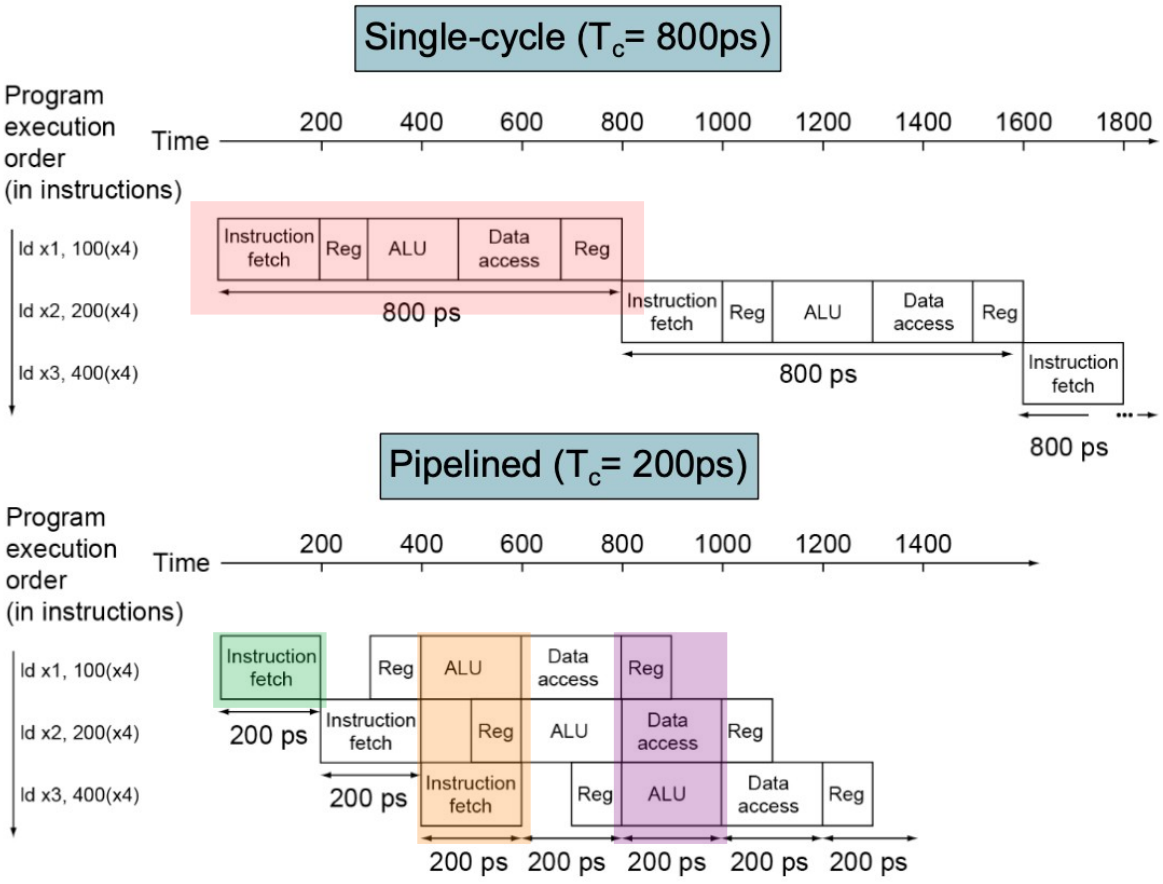
ALU/Logic	Jump/Branch	Load	Store
45%	20%	20%	15%

Tutorial Answer #1

- **Question:** What is the clock cycle time in a pipelined and non-pipelined processor?
- **Answer:**
  - Non-pipelined processor
    - Add up all the latencies together, because each stage is done sequentially, one at a time
    - So, 250 + 350 + 150 + 300 + 200 = 1250 ps
  - Pipeline processor
    - Find the greatest latency, and that's your answer
      - 350 ps

Tutorial Answer #1

Pipeline Performance

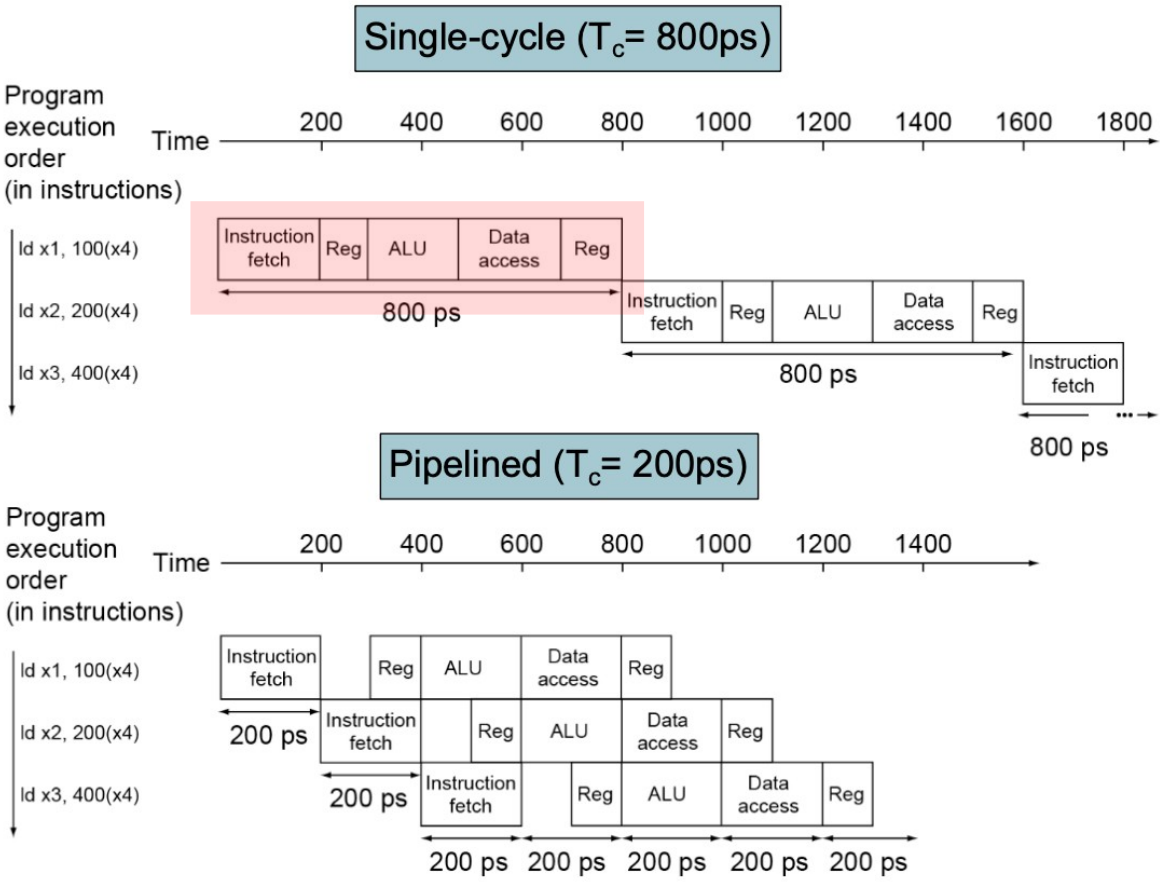


Tutorial Answer #1

- **Question:** What is the total latency of an ld instruction in a pipelined and non-pipelined processor?
- **Answer:**
  - The TOTAL latency of an ld instruction is the sum of all stages
    - 250 + 350 + 150 + 300 + 200 = 1250 ps
      - Next slide

Tutorial Answer #1

Pipeline Performance



Tutorial Answer #1

- **Question:** If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?
- **Answer:** Split the stage/step that takes the longest time
  - In this case, it is **ID** (instruction decode). Then **ID** is no longer the longest stage, **MEM** is
  - So the clock-cycle time is 300 ps for a pipelined processor

Tutorial Answer #1

- **Question:** Why not split the MEM stage as well?
- **Answer:** There's always a tradeoff. Similarly, you can't keep adding more cores to a processor. And you can't keep increasing the clock speed without encountering thermal and power issues

Tutorial Answer #1

- **Question:** Assuming there are no stalls or hazards, what is the utilization of the data memory?
- **Answer:**
  - Go back to question to get breakdown of instructions
  - Data memory:Think of load and store
    - 20 + 15 = 35%

Tutorial Answer #1

- **Question:** Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?
- **Answer:**
  - Writing to register, think of Write Register, Write Data, and RegWrite
    - What instruction writes to a register? a*dd*
      - What's being used?
        - ALU: 45%
        - Write back: 20%
  - 45 + 20 = 65%

Tutorial Question #2

- **Question:** Refer to 420.png
  - (Got lazy with the slides. Thanks Liam!)

Tutorial Question #3

- **Gonna try something different. Stay tuned…**

THE END