NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 1

Instructor Contact : ajb20@nyu.edu

Course Assistants & Graders: Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 1 [released: Tuesday Sept 6th] [due Friday Sept 16th by 11:59 PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Brightspace. *Please use the Brightspace portal to upload your completed HW.*

---

1. Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

a. Which processor has the highest performance expressed in instructions per second?

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

c. We are trying to reduce the execution time by 30%, but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

2. Assume for arithmetic, load/store, and branch instructions, a processor has CPIs of 1, 12, and 5, respectively. Also assume that on a single processor a program requires the execution of 2.56E9 **arithmetic** instructions, 1.28E9 **load/store** instructions, and 256 million **branch** instructions. Assume that each processor has a 2 GHz clock frequency.

Assume that, as the program is parallelized to run over multiple cores, the number of arithmetic and load/store instructions per processor is divided by **0.7** × **p** (where p is the number of processors) but the number of branch instructions per processor remains the same

a. Find the total execution time (ET) for this program on 1, 2, 4, and 8 processors, and show the relative speedup of the 2, 4, and 8 processors result relative to the single processor result.

b. If the CPI of the arithmetic instructions were doubled, what would the impact be on the execution time of the program on 1, 2, 4, or 8 processors?

c. To what should the CPI of load/store instructions be reduced in order for a single processor to match the performance of four processors using the original CPI values?

3. Consider the three different processors P1, P2, and P3 executing the same instruction set. P1 has a clock cycle time of 0.33 ns and CPI of 1.5; P2 has a clock cycle time of 0.40 ns and CPI of 1.0; P3 has a clock cycle time of 0.3 ns and CPI of 2.8.

a. Which has the highest clock rate? What is it?

b. Which is the fastest computer? If the answer is different than above, explain why. Which is slowest?

c. How do the answers for a and b reflect the importance of benchmarks?

4. You are designing a system for a real-time application in which specific deadlines must be met. Finishing the computation faster gains nothing. You find that your system can execute the necessary code, in the worst case, twice as fast as necessary.

a. How much energy do you save if you execute at the current speed and turn off the system when the computation is complete?
b. How much energy do you save if you set the voltage and frequency to be half as much?

5. Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D). P1 with a clock rate of 2.0 GHz and CPIs of 1, 2, 2, and 1, and P2 with a clock rate of 4 GHz and CPIs of 2, 3, 4, and 4.

a. Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D. Which is faster: P1 or P2 (in total execution time)?

b. What is the global CPI for each implementation?

c. Find the clock cycles required in both cases.

d. Which processor has the highest throughput performance (instructions per second) ?

e. Which processor do you think is more energy efficient? Why?

6. **a.** What is the difference between CISC and RISC architectures? Give some examples wherein you think CISC architectures are better suited for than RISC architectures and vice versa.

b. Describe in your own words why you think RISC V would be a better alternative compared to ARM or x86 architectures?

c. What do you think are the challenges faced by RISC V architecture going forward?

7. In this exercise, assume that we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 15 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode the *percentage of vectorization.* Vectors are discussed in Chapter 4, but you don't need to know anything about how they work to answer this question!

a. Draw a graph that plots the speedup as a percentage of the computation performed in vector mode. Label the y-axis "Net speedup" and label the x-axis "Percent vectorization."
b. What percentage of vectorization is needed to achieve a speedup of 2?
c. What percentage of the computation run time is spent in vector mode if a speedup of 2 is achieved?
d. What percentage of vectorization is needed to achieve one-half the maximum speedup attainable from using vector mode?
e. Suppose you have measured the percentage of vectorization of the program to be 70%. The hardware design group estimates it can speed up the vector hardware even more with significant additional investment. You wonder whether the compiler crew could increase the percentage of vectorization, instead. What percentage of vectorization would the compiler team need to achieve in order to equal an addition 2× speedup in the vector unit (beyond the initial 15×)?

8. In a server farm such as that used by Amazon or eBay, a single failure does not cause the entire system to crash. Instead, it will reduce the number of requests that can be satisfied at any one time.

a. If a company has 10,000 computers, each with a MTTF of 35 days, and it experiences catastrophic failure only if 1/3 of the computers fail, what is the MTTF for the system?

b. If it costs an extra $1000, per computer, to double the MTTF, would this be a good business decision? Show your work.

9. a. A program (or a program task) takes 150 million instructions to execute on a processor running at 2.7 GHz. Suppose that 70% of the instructions execute in 3 clock cycles, 20% execute in 4 clock cycles, and 10% execute in 5 clock cycles. What is the execution time for the program or task?

b. Suppose the processor in the previous question part is redesigned so that all instructions that initially executed in 5 cycles and all instructions executed in 4 cycles now execute in 2 cycles. Due to changes in the circuitry, the clock rate also must be decreased from 2.7 GHz to 1.5 GHz. What is the overall percentage improvement?

10. Availability is the most important consideration for designing servers, followed closely by scalability and throughput.

a. We have a single processor with a failure in time (FIT) of 100. What is the mean time to failure (MTTF) for this system?

b. If it takes one day to get the system running again, what is the availability of the system?

c. Imagine that the government, to cut costs, is going to build a supercomputer out of inexpensive computers rather than expensive, reliable computers. What is the MTTF for a system with 1000 processors? Assume that if one fails, they all fail.

11. Server farms such as Google and Yahoo! provide enough compute capacity for the highest request rate of the day. Imagine that most of the time these servers operate at only 60% capacity. Assume further that the power does not scale linearly with the load; that is, when the servers are operating at 60% capacity, they consume 90% of maximum power. The servers could be turned off, but they would take too long to restart in response to more load. A new system has been proposed that allows for a quick restart but requires 20% of the maximum power while in this "barely alive" state.

a. How much power savings would be achieved by turning off 60% of the servers?
b. How much power savings would be achieved by placing 60% of the servers in the "barely alive" state?
c. How much power savings would be achieved by reducing the voltage by 20% and frequency by 40%?
d. How much power savings would be achieved by placing 30% of the servers in the "barely alive" state and 30% off?

12. Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 100 million arithmetic instructions, 20 million load/store instructions, 20 million branch instructions.

a. Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can reduce the number of arithmetic instructions needed to execute a program by 25%, while increasing the clock cycle time by only 10%. Is this a good design choice? Why?
b. Suppose that we find a way to double the performance of arithmetic instructions. What is the overall speedup of our machine? What if we find a way to improve the performance of arithmetic instructions by 10 times?

NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 2

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistants

Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 2 [released Tuesday September 20th 2022] [due Friday September 30th by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

---

1. After graduating, you are asked to become the lead computer designer at Hyper Computers, Inc. Your study of usage of high-level language constructs suggests that procedure calls are one of the most expensive operations. You have invented a scheme that reduces the loads and stores normally associated with procedure calls and returns. The first thing you do is run some experiments with and without this optimization. Your experiments use the same state-of-the-art optimizing compiler that will be used with either version of the computer. These experiments reveal the following information:

- The clock rate of the unoptimized version is 5% higher.
- 30% of the instructions in the unoptimized version are loads or stores.
- The optimized version executes 2/3 as many loads and stores as the unoptimized version. For all other instructions the dynamic counts are unchanged.
- All instructions (including load and store) take one clock cycle.

Which is faster? Justify your decision quantitatively.

2. General-purpose processes are optimized for general-purpose computing. That is, they are optimized for behavior that is generally found across a large number of applications. However, once the domain is restricted somewhat, the behavior that is found across a large number of the target applications may be different from general-purpose applications. One such application is deep learning or neural networks. Deep learning can be applied to many different applications, but the fundamental building block of inference—using the learned information to make decisions—is the same across them all. Inference operations are largely parallel, so they are currently performed on graphics processing units, which are specialized more toward this type of

computation, and not to inference in particular. In a quest for more performance per watt, Google has created a custom chip using tensor processing units to accelerate inference operations in deep learning.1 This approach can be used for speech recognition and image recognition, for example. This problem explores the trade-offs between this process, a general-purpose processor (Haswell E5-2699 v3) and a GPU (NVIDIA K80), in terms of performance and cooling. If heat is not removed from the computer efficiently, the fans will blow hot air back onto the computer, not cold air. Note: The differences are more than processor—on-chip memory and DRAM also come into play. Therefore statistics are at a system level, not a chip level.

a. If Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what is the speedup of the TPU system over the GPU system?

b. Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what percentage of Max IPS does it achieve for each of the three systems?

c. Building on (b), assuming that the power scales linearly from idle to busy power as IPS grows from 0% to 100%, what is the performance per watt of the TPU system over the GPU system?

d. If another data center spends 40% of its time on workload A, 10% of its time on workload B, and 50% of its time on workload C, what are the speedups of the GPU and TPU systems over the general-purpose system?

e. A cooling door for a rack cost $4000 and dissipates 14 kW (into the room; additional cost is required to get it out of the room). How many Haswell-, NVIDIA-, or Tensor-based servers can you cool with one cooling door, assuming TDP in Figures 1.27 and 1.28?

f. Typical server farms can dissipate a maximum of 200 W per square foot. Given that a server rack requires 11 square feet (including front and back clearance), how many servers from part (e) can be placed on a single rack, and how many cooling doors are required?

System	Chip	TDP	Idle power	Busy power
General-purpose	Haswell E5-2699 v3	504 W	159 W	455 W
Graphics processor	NVIDIA K80	1838 W	357 W	991 W
Custom ASIC	TPU	861 W	290 W	384 W

Figure 1.27 Hardware characteristics for general-purpose processor, graphical processing unit-based or custom ASIC-based system, including measured power

System	Chip	Throughput			% Max IPS		
		A	B	C	A	B	C
General-purpose	Haswell E5-2699 v3	5482	13,194	12,000	42%	100%	90%
Graphics processor	NVIDIA K80	13,461	36,465	15,000	37%	100%	40%
Custom ASIC	TPU	225,000	280,000	2000	80%	100%	1%

Figure 1.28 Performance characteristics for general-purpose processor, graphical processing unit-based or custom ASIC-based system on two neural-net workloads

3. In this exercise, assume that we are considering enhancing a quad-core machine by adding encryption hardware to it. When computing encryption operations, it is 20 times faster than the normal mode of execution. We will define percentage of encryption as the percentage of time in the original execution that is spent performing encryption operations. The specialized hardware increases power consumption by 2%.

a. Draw a graph that plots the speedup as a percentage of the computation spent performing encryption. Label the y-axis "Net speedup" and label the x-axis "Percent encryption."

b. With what percentage of encryption will adding encryption hardware result in a speedup of 2?

c. What percentage of time in the new execution will be spent on encryption operations if a speedup of 2 is achieved?

4. Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 50% of the time, measured as a percentage of the execution time when the enhanced mode is in use. Recall that Amdahl's Law depends on the fraction of the original, unenhanced execution time that could make use of enhanced mode. Thus, we cannot directly use this 50% measurement to compute speedup with Amdahl's Law.

a. What is the speedup we have obtained from fast mode?

b. What percentage of the original execution time has been converted to fast mode?

5. When parallelizing an application, the ideal speedup is speeding up by the number of processors. This is limited by two things: percentage of the application that can be parallelized and the cost of communication. Amdahl's Law takes into account the former but not the latter.

a. What is the speedup with N processors if 80% of the application is parallelizable, ignoring the cost of communication?

b. What is the speedup with eight processors if, for every processor added, the communication overhead is 0.5% of the original execution time.

c. What is the speedup with eight processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?

d. What is the speedup with N processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?

e. Write the general equation that solves this question: What is the number of processors with the highest speedup in an application in which P% of the original execution time is parallelizable, and, for every time the number of processors is doubled, the communication is increased by 0.5% of the original execution time?

6. Your company has just bought a new 22-core processor, and you have been tasked with optimizing your software for this processor. You will run four applications on this system, but the resource requirements are not equal. Assume the system and application characteristics listed in Table 1.1 below (from textbook)

Table 1.1 Four applications

Application	A	B	C	D
% resources needed	41	27	18	14
% parallelizable	50	80	60	90

The percentage of resources of assuming they are all run in serial. Assume that when you parallelize a portion of the program by X, the speedup for that portion is X.

a. How much speedup would result from running application A on the entire 22-core processor, as compared to running it serially?

b. How much speedup would result from running application D on the entire 22-core processor, as compared to running it serially?

c. Given that application A requires 41% of the resources, if we statically assign it 41% of the cores, what is the overall speedup if A is run parallelized but everything else is run serially?

d. What is the overall speedup if all four applications are statically assigned some of the cores, relative to their percentage of resource needs, and all run parallelized?

e. Given acceleration through parallelization, what new percentage of the resources are the applications receiving, considering only active time on their statically-assigned cores?

7. When making changes to optimize part of a processor, it is often the case that speeding up one type of instruction comes at the cost of slowing down something else. For example, if we put in a complicated fast floating-point unit, that takes space, and something might have to be moved farther away from the middle to accommodate it, adding an extra cycle in delay to reach that unit. The basic Amdahl's Law equation does not take into account this trade-off.

a. If the new fast floating-point unit speeds up floating-point operations by, on average, 2x, and floating-point operations take 20% of the original program's execution time, what is the overall speedup (ignoring the penalty to any other instructions)?

b. Now assume that speeding up the floating-point unit slowed down data cache accesses, resulting in a 1.5x slowdown (or 2/3 speedup). Data cache accesses consume 10% of the execution time. What is the overall speedup now?

c. After implementing the new floating-point operations, what percentage of execution time is spent on floating-point operations? What percentage is spent on data cache accesses?

8. When making changes to optimize part of a processor, it is often the case that speeding up one type of instruction comes at the cost of slowing down something else. For example, if we put in a complicated fast floating-point unit, that takes space, and something might have to be moved farther away from the middle to accommodate it, adding an extra cycle in delay to reach that unit. The basic Amdahl's Law equation does not take into account this trade-off.

a. If the new fast floating-point unit speeds up floating-point operations by, on average, 2x, and floating-point operations take 20% of the original program's execution time, what is the overall speedup (ignoring the penalty to any other instructions)?

b. Now assume that speeding up the floating-point unit slowed down data cache accesses, resulting in a 1.5x slowdown (or 2/3 speedup). Data cache accesses consume 10% of the execution time. What is the overall speedup now?

c. After implementing the new floating-point operations, what percentage of execution time is spent on floating-point operations? What percentage is spent on data cache accesses?

NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 3

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistants

Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 3 [released Monday September 26th 2022] [due Wednesday October 5th by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

- *In RISCV, only load and store instructions access memory locations*
- *These instructions must follow a 'format' to access memory*
- *Assume a 32-bit machine in all problems unless asked to assume otherwise*

Problem 1:

Assume address in memory of '`A[0]`', '`B[0]`' and '`C[0]`') are stored in Registers `x27, x30, x31`. Assume values of variables `f, g, h, i,` and `j` are assigned to registers `x5, x6, x7, x28, x29` respectively

Write down RISC V Instruction(s) to

(a) Load Register `x5` *with content* of `A[10]`

(b) Store contents of Register `x5` into `A[17]`

(c) add 2 operands: one in `x5` - a register, the other in in Register x6. Assume result of operation to be stored in register `x7`

(d) copy contents at one memory location to another: `C[g] = A[i+j+31]`

(e) implement in RISC V these line of code in C:

(i) `f = g - A[B[9]]`

(ii) `f = g - A[C[8] + B[4]]`

(iii) `A[i] = B[2i+1], C[i] = B[2i]`

(iv) `A[i] = 4B[i-1] + 4C[i+1]`

(v) `f = g - A[C[4] + B[12]]`

Problem 2:

Assume the following register contents:

`x5 = 0x00000000AAAAAAAA, x6 = 0x1234567812345678`

a. For the register values shown above, what is the value of `x7` for the following sequence of instructions?

```
srli x7, x5, 16
addi x7, x7, -128
srai x7, x7, 2
and x7, x7, x6
```

b. For the register values shown above, what is the value of `x7` for the following sequence of instructions?

```
slli x7, x6, 4
```

c. For the register values shown above, what is the value of `x7` for the following sequence of instructions?

```
srli x7, x5, 3
andi x7, x7, 0xFEF
```

Problem 3:

For each RISC-V instruction below, identify the instruction format and show, wherever applicable, the value of the opcode (`op`), source register (`rs1`), source register (`rs2`), destination register (`rd`), immediate (`imm`), `func3`, `func7` fields. Also provide the 8 hex char (or 32 bit) instruction for each of the instructions below

```
add x5, x6, x7
addi x8, x5, 512
ld x3, 128(x27)
sd x3, 256(x28)
beq x5, x6 ELSE  #ELSE is the label of an instruction 16 bytes larger
                 #than the current content of PC
```

```
add x3, x0, x0
auipc x3, FFEFA
jal x3 ELSE
```

Problem 4:

(a) For the following C statement, write a minimal sequence of RISC-V assembly instructions that performs the identical operation. Assume `x5 = A`, and `x11` is the base address of C.

```
A = C[0] << 16;
```

(b) Find the shortest sequence of RISC-V instructions that extracts `bits 12 down to 7` from register `x3` and uses the value of this field to replace bits `28 down to 23` in register `x4` without changing the other bits of registers `x3` or `x4`. (Be sure to test your code using `x3 = 0` and `x4 = 0xffffffffffffffff`. Doing so may reveal a common oversight.)

(c) Provide a minimal set of RISC-V instructions that may be used to implement the following pseudoinstruction:

```
not x5, x6 // bit-wise invert
```

[Hint: note that there is no 'not' instruction in RISCV. However, an XOR immediate instruction could be used]

Problem 5:

Suppose the program counter (PC) is set to $0x60000000_{hex}$.

a. What range of addresses can be reached using the RISC-V *jump-and-link* (`jal`) instruction? (In other words, what is the set of possible values for the PC after the jump instruction executes?)

b. What range of addresses can be reached using the RISC-V *branch if equal* (`beq`) instruction? (In other words, what is the set of possible values for the PC after the branch instruction executes?)

Problem 6:

Assume that the register `x6` is initialized to the value 10. What is the final value in register `x5` assuming the `x5` is initially zero?

```
LOOP:     beq x6, x0, DONE
          addi x6, x6, -1
          addi x5, x5, 2
          jal x0, LOOP
DONE:
```

a. For the loop above, write the equivalent C code. Assume that the registers `x5` and `x6` are integers `acc` and `i`, respectively.

b. For the loop written in RISC-V assembly above, assume that the register `x6` is initialized to the value N. How many RISC-V instructions are executed?

c. For the loop written in RISC-V assembly above, replace the instruction "`beq x6, x0, DONE`" with the instruction "`blt x6, x0, DONE`" and write the equivalent C code.

Problem 7:

a. Translate the following C code to RISC-V assembly code. Use a minimum number of instructions. Assume that the values of `a, b, i,` and `j` are in registers `x5, x6, x7,` and `x29,` respectively. Also, assume that register `x10` holds the base address of the array `D`.

```
for(i=0; i<a; i++)
     for(j=0; j<b; j++)
          D[4*j] = i + j;
```

b. How many RISC-V instructions does it take to implement the C code from 7a. above? If the variables **a** and **b** are initialized to **10** and **1** and all elements of **D** are initially 0, what is the total number of RISC-V instructions executed to complete the loop?

Problem 8:

Consider the following code:

```
lb x6, 0(x7)
sd x6, 8(x7)
```

Assume that the register x7 contains the address **0×10000000** and the data at address is 0×**11**22334455667**88**.

a. What value is stored in `0×10000007` on a bigendian machine?

b. What value is stored in **0×10000007** on a littleendian machine?

Problem 9:

Write the RISC-V assembly code that creates the 64-bit constant $0x1234567812345678_{hex}$ and stores that value to register `x10`.

Problem 10: Assume that **x5** holds the value $\mathbf{128_{10}}$.

a. For the instruction **add x30, x5, x6,** what is the range(s) of values for **x6** that would result in overflow?

b. For the instruction **sub x30, x5, x6,** what is the range(s) of values for **x6** that would result in overflow?

c. For the instruction **sub x30, x6, x5,** what is the range(s) of values for **x6** that would result in overflow?

NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 4

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistants

Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 4 [released Wednesday October 5th 2022] [due Wednesday October 12th by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

---

1. How would you test for overflow, the result of an addition of two 8-bit operands if the operands were (i) unsigned (ii) signed with 2s complement representation.

Add the following 8-bit strings assuming they are (i) *unsigned* (ii) *signed and represented using 2's complement*. Indicate *which of these additions overflow*.

A. 0110 1110 + 1001 1111

B. 1111 1111 + 0000 0001

C. 1000 0000 + 0111 1111

D. 0111 0001 + 0000 1111

2. One possible performance enhancement is to do a shift and add instead of an actual multiplication. Since 9×6, for example, can be written (2×2×2+1)×6, we can calculate 9×6 by shifting 6 to the left three times and then adding 6 to that result. Show the best way to calculate $0xAB_{hex} \times 0xEF_{hex}$ using shifts and adds/subtracts. Assume both inputs are 8-bit unsigned integers.

3. What decimal number does the 32-bit pattern 0×DEADBEEF represent if it is a floating-point number? Use the IEEE 754 standard

4. Write down the binary representation of the decimal number 78.75 assuming the IEEE 754 *single precision* format. Write down the binary representation of the decimal number 78.75 assuming the IEEE 754 *double precision* format

5. Write down the binary representation of the decimal number `78.75` assuming it was stored using the single precision **IBM format** (base 16, instead of base 2, with 7 bits of exponent).

6. `IEEE 754-2008` contains a half precision that is only `16 bits` wide. The leftmost bit is still the sign bit, the exponent is `5 bits` wide and has a `bias of 15`, and the mantissa (fractional field) is `10 bits` long. A hidden 1 is assumed.
(a) Write down the bit pattern to represent **-1.3625×10^{-1}** Comment on how the range and accuracy of this 16-bit floating point format compares to the single precision `IEEE 754` standard.

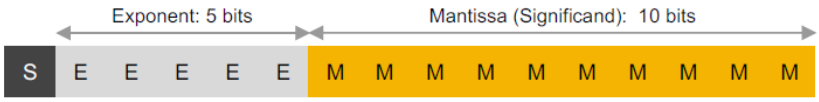
(b) Calculate the sum of **1.6125×10^{1}** `(A)` and **$3.150390625 \times 10^{-1}$** `(B)` by hand, assuming operands A and B are stored in the 16- bit half precision described in problem a. above Assume `1 guard, 1 round bit, and 1 sticky bit`, and round to the nearest even. Show all the steps.

7. What is the range of representation and relative accuracy of positive numbers for the following 3 formats:

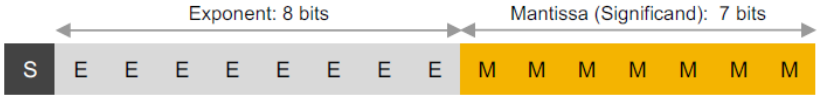
(i) `IEEE 754` Single Precision (ii) `IEEE 754 – 2008` (described in Problem 6 above) and (iii) `'bfloat16'` shown in the figure below

IEEE 754, Single Precision:

fp16: Half-precision IEEE Floating Point Format



bfloat16: Brain Floating Point Format



8. Suppose we have a 7-bit computer that uses IEEE floating-point arithmetic where a floating point number has 1 sign bit, 3 exponent bits, and 3 fraction bits. All of the bits in the hardware work properly.

Recall that denormalized numbers will have an exponent of 000, and the `bias` for a 3-bit exponent is

$2^{3-1} - 1 = 3$.

(a) For each of the following, write the *binary value* and *the corresponding decimal value* of the 7-bit floating point number that is the closest available representation of the requested number. If rounding is necessary use round-to-nearest. Give the decimal values either as whole numbers or fractions. The first few lines are filled in for you.

Number	Binary	Decimal
0	0 000 000	0.0
-0.125	1 000 000	-0.125
Smallest positive normalized number		
largest positive normalized number		
Smallest positive denormalized number > 0		
largest positive denormalized number > 0		

(b) The associative law for addition says that a + (b + c) = (a + b) + c. This holds for regular arithmetic, but does not always hold for floating-point numbers. Using the 7-bit floating-point system described above, give an example of three floating-point numbers a, b, and c for which the associative law does not hold, and show why the law does not hold for those three numbers.

NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 5

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistants

Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 4 [released Wednesday October 23rd 2022] [due Wednesday November 2nd by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

.

---

1. In this exercise, we examine in detail how an instruction is executed in a single-cycle datapath. Problems in this exercise refer to a clock cycle in which the processor fetches the following instruction word: **0x00c6ba23**.

1.1 What are the values of the ALU control unit's inputs for this instruction?

1.2 What is the new PC address after this instruction is executed? Highlight the path through which this value is determined.

1.3 For each mux, show the values of its inputs and outputs during the execution of this instruction. List values that are register outputs at Reg [xn].

1.4 What are the input values for the ALU and the two add units?

1.5 What are the values of all inputs for the register's unit?

2. Problems in this exercise assume that the logic blocks used to implement a processor's datapath have the following latencies:

I-Mem / D-Mem	Register File	Mux	ALU	Adder	Single gate	Register Read	Register Setup	Sign extend	Control
250 ps	150 ps	25 ps	200 ps	150 ps	5 ps	30 ps	20 ps	50 ps	50 ps

"Register read" is the time needed after the rising clock edge for the new register value to appear on the output. This value applies to the PC only. "Register setup" is the amount of time a register's data input must be stable before the rising edge of the clock. This value applies to both the PC and Register File.

2.1 What is the latency of an `R-type` instruction (i.e., how long must the clock period be to ensure that this instruction works correctly)?
2.2 What is the latency of `ld`? (Check your answer carefully. Many students place extra muxes on the critical path.)
2.3 What is the latency of `sd`? (Check your answer carefully. Many students place extra muxes on the critical path.)
2.4 What is the latency of `beq`?
2.5 What is the latency of an I-type instruction?
2.6 What is the minimum clock period for this CPU?

3. (a) Suppose you could build a CPU where the clock cycle time was different for each instruction.

3.a1 What would the speedup of this new CPU be over the CPU presented in Figure 4.21 (in RISC-V text) given the instruction mix below? (assuming instruction latencies from the problem 2)

R-type/I-type (non-ld)	ld	sd	beq
52%	25%	11%	12%

3 (b) Consider the addition of a multiplier to the CPU shown in Figure 4.21. This addition will add 300 ps to the latency of the ALU, but will reduce the number of instructions by 5% (because there will no longer be a need to emulate the multiply instruction).

3.b1 What is the clock cycle time with and without this improvement?

3.b2 What is the speedup achieved by adding this improvement?

3.b3 What is the slowest the new ALU can be and still result in improved performance?

3 (c) When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are beginning with the datapath from Figure 4.21, the latencies from **`Problem 2`** in this assignment, and the following costs:

I-Mem	Register File	Mux	ALU	Adder	D-Mem	Single Register	Sign extend	Single gate	Control
1000	200	10	100	30	2000	5	100	1	500

Suppose doubling the number of general-purpose registers from 32 to 64 would reduce the number of `ld` and `sd` instruction by 12%, but increase the latency of the register file from 150 ps to 160 ps and double the cost from 200 to 400. (*Use the instruction mix [from 3(a) above] and ignore the other effects on the ISA*)

3.c1 What is the speedup achieved by adding this improvement?
3.c2 Compare the change in performance to the change in cost.
3.c3 Given the cost/performance ratios you just calculated, describe a situation where it makes sense to add more registers and describe a situation where it doesn't make sense to add more registers.

4. `ld` is the instruction with the longest latency on the CPU from Section 4.4 (in RISC-V text). If we modified `ld` and sd so that there was no offset (i.e., the address to be loaded from/stored to must be calculated and placed in `rs1` before calling `ld/sd`), then no instruction would use both the ALU and Data memory. This would allow us to reduce the clock cycle time. However, it would also increase the number of instructions, because many `ld` and `sd` instructions would need to be replaced with `ld/add` or `sd/add` combinations.

4.1 What would the new clock cycle time be?

4.2 Would a program with the instruction mix presented *in Problem 2* run faster or slower on this new CPU? By how much? (For simplicity, assume every ld and sd instruction is replaced with a sequence of two instructions.)

4.3 What is the primary factor that influences whether a program will run faster or slower on the new CPU?

4.4 Do you consider the original CPU (*as shown in Figure 4.21 of RISC-V text*) a better overall design; or do you consider the new CPU a better overall design? Why?

5. (a) Examine the difficulty of adding a proposed `lwi.d rd, rs1, rs2` ("Load With Increment") instruction to RISC-V. Interpretation: `Reg[rd]=Mem[Reg[rs1]+Reg[rs2]]`

5.a1 Which new functional blocks (if any) do we need for this instruction?

5.a2 Which existing functional blocks (if any) require modification?

5.***a3*** Which new data paths (if any) do we need for this instruction?

6. 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%

6.1 What is the clock cycle time in a pipelined and non-pipelined processor?
6.2 What is the total latency of an `ld` instruction in a pipelined and non-pipelined processor?
6.3 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?
6.4 Assuming there are no stalls or hazards, what is the utilization of the data memory?
6.5 Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

7. What is the minimum number of cycles needed to completely execute n instructions on a CPU with a k stage pipeline? Justify your formula.

8. (a) Assume that `x11` is initialized to `11` and `x12` is initialized to `22`. Suppose you executed the code below on a version of the pipeline from Section 4.5 that does not handle data hazards (i.e., the programmer is responsible for addressing data hazards by inserting NOP instructions where necessary). What would the final values of registers `x13` and `x14` be?

```
addix11, x12, 5

addx13, x11, x12

addix14, x11, 15
```

(b) Assume that `x11` is initialized to `11` and `x12` is initialized to `22`. Suppose you executed the code below on a version of the pipeline from Section 4.5 *that does not handle data hazards* (i.e., the programmer is responsible for addressing data hazards by inserting NOP instructions where necessary).

What would the final values of register `x15` be? Assume the register file is written at the beginning of the cycle and read at the end of a cycle. Therefore, an `ID` stage will return the results of a `WB` state occurring during the same cycle. See Section 4.7 and Figure 4.51 for details.

```
addix11, x12, 5
addx13, x11, x12
addix14, x11, 15
addx15, x11, x11
```

(c) Add `NOP` instructions to the code below so that it will run correctly on a pipeline that does not handle data hazards.

```
addix11, x12, 5
addx13, x11, x12
addix14, x11, 15
addx15, x13, x12
```

NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 6

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistants

Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 6 [released Friday November 4th 2022] [due Friday November 11th by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

---

1. Consider a version of the pipeline from *Section 4.5 in RISC-V text* that does not handle data hazards (i.e., the programmer is responsible for addressing data hazards by inserting NOP instructions where necessary). Suppose that (after optimization) a typical n- instruction program requires an additional **0.**4*n NOP instructions to correctly handle data hazards.

1.1 Suppose that the cycle time of this pipeline without forwarding is 250 ps. Suppose also that adding forwarding hardware will reduce the number of NOPs from .4*n to .05*n, but increase the cycle time to 300 ps. What is the speedup of this new pipeline compared to the one without forwarding?

1.2 Different programs will require different amounts of NOPs. How many NOPs (as a percentage of code instructions) can remain in the typical program before that program runs slower on the pipeline with forwarding?

1.3 Repeat 1.2; however, this time let x represent the number of NOP instructions relative to n. (In 1.2, x was equal to 0.4) Your answer will be with respect to x.

1.4 Can a program with only .075*n NOPs possibly run faster on the pipeline with forwarding? Explain why or why not.

1.5 At minimum, how many NOPs (as a percentage of code instructions) must a program have before it can possibly run faster on the pipeline with forwarding?

2. Consider the fragment of RISC-V assembly below:

```
sd x29, 12(x16)
ld x29, 8(x16)
sub x17, x15, x14
beqz x17, label
add x15, x11, x14
sub x15, x30, x14
```

Suppose we modify the pipeline so that it has only one memory (*that handles both instructions and data*). In this case, there will be a structural hazard every time a program needs to fetch an instruction during the same cycle in which another instruction accesses data.

2.1 Draw a pipeline diagram to show were the code above will stall.

2.2 In general, is it possible to reduce the number of stalls/`NOPs` resulting from this structural hazard by reordering code?

2.3 Must this structural hazard be handled in hardware? We have seen that data hazards can be eliminated by adding `NOPs` to the code. Can you do the same with this structural hazard? If so, explain how. If not, explain why not.

2.4 Approximately how many stalls would you expect this structural hazard to generate in a typical program? (*Use the instruction mix shown below*)

R-type/I-type (non-ld)	ld	sd	beq
52%	25%	11%	12%

3. If we change load/store instructions to use a register (without an offset) as the address, these instructions no longer need to use the `ALU`. (See Problem 4 in HW 4) As a result, the `MEM` and `EX` stages can be overlapped and the pipeline has only four stages.

3.1 How will the reduction in pipeline depth affect the cycle time?

3.2 How might this change improve the performance of the pipeline?

3.3 How might this change degrade the performance of the pipeline?

4. Which of the two pipeline diagrams below better describes the operation of the pipeline's hazard detection unit? Why?

Choice 1:

```
ld x11, 0(x12): IF ID EX ME WB
add x13, x11, x14: IF ID EX..ME WB
or x15, x16, x17: IF ID..EX ME WB
```

Choice 2:

```
ld x11, 0(x12): IF ID EX ME WB
add x13, x11, x14: IF ID..EX ME WB
or x15, x16, x17: IF..ID EX ME WB
```

5. Consider the following loop.

```
LOOP: ld   x10, 0(x13)
      ld   x11, 8(x13)
      add  x12, x10, x11
      subi x13, x13, 16
      bnez x12, LOOP
```

Assume that perfect branch prediction is used (no stalls due to control hazards), that there are no delay slots, that the pipeline has full forwarding support, and that branches are resolved in the `EX` (as opposed to the `ID`) stage.

5.1 Show a pipeline execution diagram for the first two iterations of this loop.

`Please see below in response to 5.2`

5.2 Mark pipeline stages that do not perform useful work. How often while the pipeline is full do we have a cycle in which all five pipeline stages are doing useful work? (Begin with the cycle during which the `subi` is in the `IF` stage. End with the cycle during which the `bnez` is in the `IF` stage.)

6. This exercise is intended to help you understand the cost/complexity/performance trade-offs of forwarding in a pipelined processor. Problems in this exercise refer to pipelined datapaths from *Figure 4.53 in RISC-V text (reproduced below)*. These problems assume that, of all the instructions executed in a processor, the following fraction of these instructions has a particular type of RAW data dependence.

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

The type of `RAW` data dependence is identified by the stage that produces the result (`EX` or `MEM`) and the next instruction that consumes the result (1st instruction that follows the one that produces the result, 2nd instruction that follows, or both). We assume that the register write is done in the first half of the clock cycle and that register reads are done in the second half of the cycle, so "`EX` to 3rd" and "`MEM` to 3rd" dependences are not counted because they cannot result in data hazards. We also assume that branches are resolved in the `EX` stage (as opposed to the `ID` stage), and that the CPI of the processor is 1 if there are no data hazards.

EX to 1st Only	MEM to 1st Only	EX to 2nd Only	MEM to 2nd Only	EX to 1st and EX to 2nd
5%	20%	5%	10%	10%

Assume the following latencies for individual pipeline stages. For the `EX` stage, latencies are given separately for a processor without forwarding and for a processor with different kinds of forwarding.

IF	ID	EX (no FW)	EX (full FW)	EX (FW from EX/MEM only)	EX (FW from MEM/WB only)	MEM	WB
120 ps	100 ps	110 ps	130 ps	120 ps	120 ps	120 ps	100 ps

6.1 For each RAW dependency listed above, give a sequence of at least three assembly statements that exhibits that dependency.

6.2 For each RAW dependency above, how many NOPs would need to be inserted to allow your code from ***6.1*** to run correctly on a pipeline with no forwarding or hazard detection? Show where the NOPs could be inserted.

6.3 Analyzing each instruction independently will over-count the number of NOPs needed to run a program on a pipeline with no forwarding or hazard detection. Write a sequence of three assembly instructions so that, when you consider each instruction in the sequence independently, the sum of the stalls is larger than the number of stalls the sequence actually needs to avoid data hazards.

6.4 Assuming no other hazards, what is the CPI for the program described by the table above when run on a pipeline with no forwarding? What percent of cycles are stalls? (For simplicity, assume that all necessary cases are listed above and can be treated independently.)

EX to 1st Only	MEM to 1st Only	EX to 2nd Only	MEM to 2nd Only	EX to 1st and EX to 2nd
5%	20%	5%	10%	10%

6.5 What is the CPI if we use full forwarding (forward all results that can be forwarded)? What percent of cycles are stalls?

6.6 Let us assume that we cannot afford to have three-input multiplexors that are needed for full forwarding. We have to decide if it is better to forward only from the EX/MEM pipeline register (next-cycle forwarding) or only from the MEM/WB pipeline register (two-cycle forwarding). What is the CPI for each option?

6.7 For the given hazard probabilities and pipeline stage latencies, what is the speedup achieved by each type of forwarding (EX/MEM, MEM/WB, for full) as compared to a pipeline that has no forwarding?

6.8 What would be the additional speedup (relative to the fastest processor from 6.7) be if we added "timetravel" forwarding that eliminates all data hazards? Assume that the yet-to-be-invented time-travel circuitry adds 100 ps to the latency of the full-forwarding EX stage.

7. Problems in this exercise refer to the following sequence of instructions, and assume that it is executed on a five-stage pipelined datapath:

```
add x15, x12, x11
ld x13, 4(x15)
ld x12, 0(x2)
or x13, x15, x13
sd x13, 0(x15)
```

7.1 If there is no forwarding or hazard detection, insert `NOPs` to ensure correct execution.

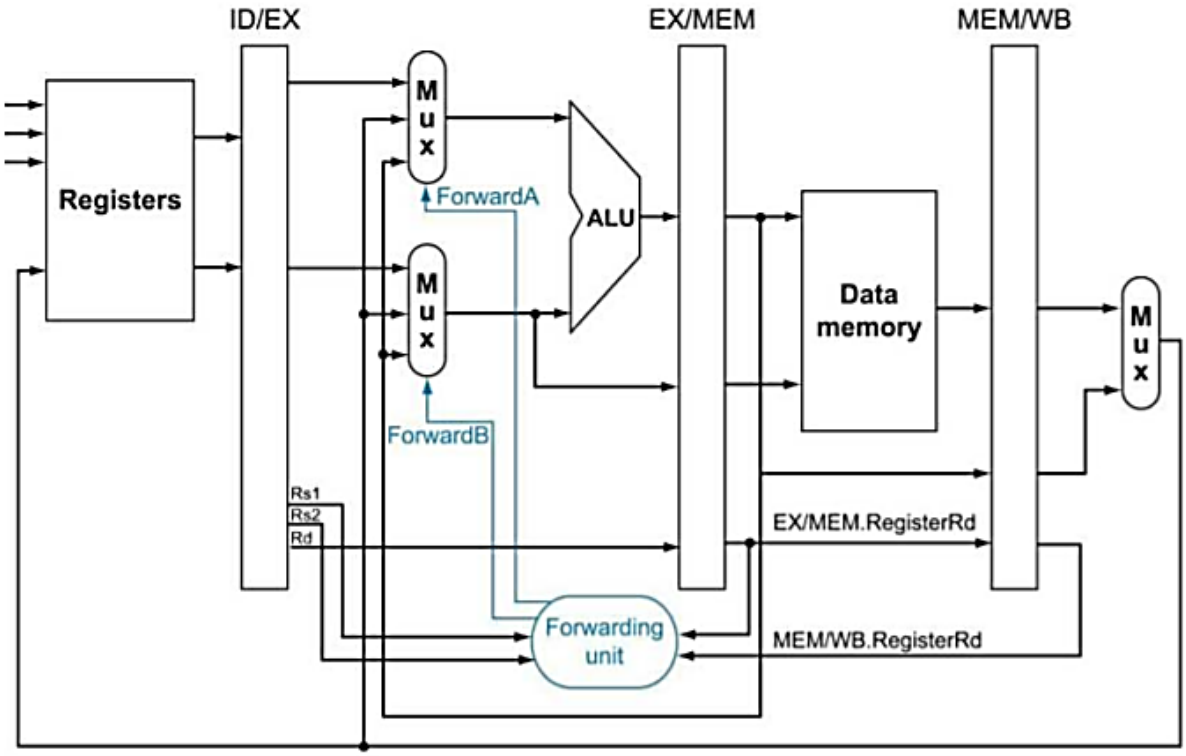
7.2 Now, change and/or rearrange the code to minimize the number of `NOPs` needed. You can assume register `x17` can be used to hold temporary values in your modified code.

7.3 If the processor has forwarding, but we forgot to implement the hazard detection unit, what happens when the original code executes?

7.4 If there is forwarding, for the first seven cycles during the execution of this code, specify which signals are asserted in each cycle by hazard detection and forwarding units in Figure 4.53 of the RISC V text (reproduced below).

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

Clock Cycle	1	2	3	4	5	6	7	8	9
add	IF	ID	EX	MEM	WB				
ld		IF	ID	EX	MEM	WB			
ld			IF	ID	EX	MEM	WB		
or				IF	ID	EX	MEM	WB	
sd					IF	ID	EX	MEM	WB



b. With forwarding

7.5 If there is no forwarding, what new input and output signals do we need for the hazard detection unit in the Figure above? Using this instruction sequence as an example, explain why each signal is needed.

7.6 For the new hazard detection unit from Problem 6.5 of this HW assignment, specify which output signals it asserts in each of the first five cycles during the execution of this code.

Clock Cycle	1	2	3	4	5	6	7	8	9
add	IF	ID	EX	MEM	WB				
ld		IF	ID	-	-	EX	MEM	WB	
ld			IF	-	-	ID	EX	MEM	WB

NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 7

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistants

Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 7 [released Friday November 11th 2022] [due Monday November 28th by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

.

---

1. Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 64-bit memory address references, given as word addresses.

`0x02, 0xb3, 0x2a, 0x01, 0xbe, 0x57, 0xbf, 0x0d, 0xb6, 0x2b, 0xbc, 0xfd`

1.1 For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list whether each reference is a hit or a miss, assuming the cache is initially empty.

1.2 For each of these references, identify the binary word address, the tag, the index, and the offset given a direct-mapped cache with two-word blocks and a total size of eight blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

1.3 You are asked to optimize a cache design for the given references. There are three direct-mapped cache designs possible, all with a total of eight words of data:
C1 has 1-word blocks,
C2 has 2-word blocks, and
C3 has 4-word blocks.

2. *Section 5.3* shows the typical method to index a direct-mapped cache, specifically (Block address) modulo (Number of blocks in the cache). Assuming a 64-bit address and 1024 blocks in the cache, consider a different indexing function, specifically (Block address[63:54] XOR Block address[53:44]). Is it possible to use this to index a direct-mapped cache? If so, explain why and discuss any changes that might need to be made to the cache. If it is not possible, explain why.

3. For a direct-mapped cache design with a 64-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
63–10	9–5	4–0

3.1 What is the cache block size (in words)?

3.2 How many blocks does the cache have?

3.3 What is the ratio between total bits required for such a cache implementation over the data storage bits?

Beginning from power on, the following byte-addressed cache references are recorded.

Address												
Hex	00	04	10	84	E8	A0	400	1E	8C	C1C	B4	884
Dec	0	4	16	132	232	160	1024	30	140	3100	180	2180

3.4 For each reference, list (1) its tag, index, and offset, (2) whether it is a hit or a miss, and (3) which bytes were replaced (if any).

3.5 What is the hit ratio?

3.6 List the final state of the cache, with each valid entry represented as a record of <*index, tag, data*>. For example,

<0, 3, Mem[oxCoo]-Mem[oxC1F]>

4. Recall that we have two write policies and two write allocation policies, and their combinations can be implemented either in L1 or L2 cache. Assume the following choices for L1 and L2 caches:

L1	L2
Write through, non-write allocate	Write back, write allocate

4.1 Buffers are employed between different levels of memory hierarchy to reduce access latency. For this given configuration, list the possible buffers needed between L1 and L2 caches, as well as L2 cache and memory.

4.2 Describe the procedure of handling an L1 write-miss, considering the components involved and the possibility of replacing a dirty block.

4.3 For a multilevel exclusive cache configuration (a block can only reside in one of the L1 and L2 caches), describe the procedures of handling an L1 write-miss and an L1 read-miss, considering the components involved and the possibility of replacing a dirty block.

5. Consider the following program and cache behaviors.

Data Reads per 1000 Instructions	Data Writes per 1000 Instructions	Instruction Cache Miss Rate	Data Cache Miss Rate	Block Size (bytes)
250	100	0.30%	2%	64

5.1 Suppose a CPU with a write-through, writeallocate cache achieves a CPI of 2. What are the read and write bandwidths (measured by bytes per cycle) between RAM and the cache? (Assume each miss generates a request for one block.)

5.2 For a write-back, write-allocate cache, assuming 30% of replaced data cache blocks are dirty, what are the read and write bandwidths needed for a CPI of 2?

6. Media applications that play audio or video files are part of a class of workloads called "streaming" workloads (i.e., they bring in large amounts of data but do not reuse much of it). Consider a video streaming workload that accesses a 512 KiB working set sequentially with the following word address stream:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 …

6.1 Assume a 64 KiB direct-mapped cache with a 32-byte block. What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C model?

6.2 Re-compute the miss rate when the cache block size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?

6.3 "*Prefetching*" is a technique that leverages predictable address patterns to speculatively bring in additional cache blocks when a particular cache block is accessed. One example of prefetching is a stream buffer that prefetches sequentially adjacent cache blocks into a separate buffer when a particular cache block is brought in. If the data are found in the prefetch buffer, it is considered as a hit, moved into the cache, and the next cache block is prefetched. Assume a two-entry stream buffer; and, assume that the cache latency is such that a cache block can be loaded before the computation on the previous cache block is completed. What is the miss rate for the address stream above?

7. Cache block size (B) can affect both miss rate and miss latency. Assuming a machine with a base CPI of 1, and an average of 1.35 references (both instruction and data) per instruction, find the block size that minimizes the total miss latency given the following miss rates for various block sizes.

8: 4%	16: 3%	32: 2%	64: 1.5%	128: 1%

7.1 What is the optimal block size for a miss latency of 20 ×B cycles?

7.2 What is the optimal block size for a miss latency of 24 +B cycles?

7.3 For constant miss latency, what is the optimal block size?

8. In this exercise, we will look at the different ways capacity affects overall performance. In general, cache access time is proportional to capacity. Assume that main memory accesses take 70 ns and that 36% of all instructions access data memory. The following table shows data for L1 caches attached to each of two processors, P1 and P2.

	L1 Size	L1 Miss Rate	L1 Hit Time
P1	2 KiB	8.0%	0.66 ns
P2	4 KiB	6.0%	0.90 ns

8.1 Assuming that the L1 hit time determines the cycle times for P1 and P2, what are their respective clock rates?

8.2 What is the Average Memory Access Time for P1 and P2 (in cycles)?

8.3 Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 and P2? Which processor is faster? (When we say a "base CPI of 1.0", we mean that instructions complete in one cycle, unless either the instruction access or the data access causes a cache miss.)

we will now consider the addition of an L2 cache to P1 (to presumably make up for its limited L1 cache capacity). Use the L1 cache capacities and hit times from the previous table when solving these problems. The L2 miss rate indicated is its local miss rate.

L2 Size	L2 Miss Rate	L2 Hit Time
1 MiB	95%	5.62 ns

8.4 What is the AMAT for P1 with the addition of an L2 cache? Is the AMAT better or worse with the L2 cache?

8.5 Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 with the addition of an L2 cache?

8.6 What would the L2 miss rate need to be in order for P1 with an L2 cache to be faster than P1 without an L2 cache?

8.7 What would the L2 miss rate need to be in order for P1 with an L2 cache to be faster than P2 without an L2 cache?

9. This exercise examines the effect of different cache designs, specifically comparing associative caches to the direct-mapped caches from *Section 5.4*. For these exercises, refer to the sequence of word address shown below.

0x03, 0xb4, 0x2b, 0x02, 0xbe, 0x58, 0xbf, 0x0e, 0x1f, 0xb5, 0xbf, 0xba, 0x2e, 0xce

9.1 Sketch the organization of a three-way set associative cache with two-word blocks and a total size of 48 words. Your sketch should have a style similar to *Figure 5.18*, but clearly show the width of the tag and data fields.

9.2 Trace the behavior of the cache from Exercise 9.1 Assume a true LRU replacement policy. For each reference, identify

- *the binary word address,*
- *the tag,*
- *the index,*
- *the offset*
- *whether the reference is a hit or a miss, and*
- *which tags are in each way of the cache after the reference has been handled.*

9.3 Sketch the organization of a fully associative cache with one-word blocks and a total size of eight words. Your sketch should have a style similar to Figure 5.18, but clearly show the width of the tag and data fields.

9.4 Trace the behavior of the cache from Exercise 9.3. Assume a true LRU replacement policy. For each reference, identify

- *the binary word address,*
- *the tag,*
- *the index,*
- *the offset,*
- *whether the reference is a hit or a miss*
- *the contents of the cache after each reference has been handled.*

9.5 Sketch the organization of a fully associative cache with two-word blocks and a total size of eight words. Your sketch should have a style similar to Figure 5.18, but clearly show the width of the tag and data fields.

9.6 Trace the behavior of the cache from Exercise 9.5. Assume an LRU replacement policy. For each reference, identify

- *the binary word address,*
- *the tag,*
- *the index,*
- *the offset,*
- *whether the reference is a hit or a miss,*
- *the contents of the cache after each reference has been handled.*

9.7 Repeat Exercise 9.6 using MRU (most recently used) replacement.

9.8 Repeat Exercise 9.6 using the optimal replacement policy (i.e., the one that gives the lowest miss rate).

10. Multilevel caching is an important technique to overcome the limited amount of space that a first-level cache can provide while still maintaining its speed. Consider a processor with the following parameters:

Base CPI, No Memory Stalls	Processor Speed	Main Memory Access Time	First-Level Cache Miss Rate per Instruction**	Second-Level Cache, Direct-Mapped Speed	Miss Rate with Second-Level Cache, Direct-Mapped	Second-Level Cache, Eight-Way Set Associative Speed	Miss Rate with Second-Level Cache, Eight-Way Set Associative
1.5	2 GHz	100 ns	7%	12 cycles	3.5%	28 cycles	1.5%

***First Level Cache miss rate is per instruction. Assume the total number of L1 cache misses (instruction and data combined) is equal to 7% of the number of instructions.*

10.1 Calculate the CPI for the processor in the table using: 1) only a first-level cache, 2) a second-level direct mapped cache, and 3) a second-level eight-way set associative cache. How do these numbers change if main memory access time doubles? (Give each change as both an absolute CPI and a percent change.) Notice the extent to which an L2 cache can hide the effects of a slow memory.

10.2 It is possible to have an even greater cache hierarchy than two levels? Given the processor above with a second-level, direct-mapped cache, a designer wants to add a third-level cache that takes 50 cycles to access and will have a 13% miss rate. Would this provide better performance? In general, what are the advantages and disadvantages of adding a third-level cache?

10.3 In older processors, such as the Intel Pentium or Alpha 21264, the second level of cache was external (located on a different chip) from the main processor and the first-level cache. While this allowed for large second-level caches, the latency to access the cache was much higher, and the bandwidth was typically lower because the second-level cache ran at a lower frequency. Assume a 512 KiB off-chip second level cache has a miss rate of 4%. If each additional 512 KiB of cache lowered miss rates by 0.7%, and the cache had a total access time of 50 cycles, how big would the cache have to be to match the performance of the second-level direct-mapped cache listed above?