CDT204 – Computer Architecture

> Date: Dec. 16th 2022
>
> Time: 8:30 – 12:30
>
> Help: Small calculator is allowed in the exam

The exam has 35 points and 3 Bonus points. The grades will be awarded as follows:

- 3 : 18 Points
- 4 : 24 Points
- 5 : 31 Points

Important Notes:

- Give as full an answer as possible to obtain full marks. All calculations, approximations, assumptions and justifications must be reported for full credit unless stated otherwise. Please use figures and examples to clarify.
- If you do not understand a question clearly, you are allowed to call the teacher and ask.
- Write the question and part number on each page clearly.
- Answer each question on a separate page.
- In case you might have forgotten:

  $1GB = 2^{10}MB = 2^{20}KB = 2^{30}B$

  $1\ sec = 10^{3}\ ms = 10^{6}\ \mu s = 10^{9}\ ns = 10^{12}\ ps$

Live long and prosper

Task 1 – General (4p)

A. How do you do subtraction with two's complement? (Explain in words). (1p)
B. For a 4-bit two's complement number, show how the following operations are represented and their results. If you cannot, explain why. (3p)
    a. 4 – 7
    b. 11 – 3
    c. -4 - 4

Task 2 – Performance (6p)

When a program is adapted to run on multiple processors in a multiprocessor system, the execution time on each processor consists of computing time and the overhead time required for locked critical sections and/or to send data from one processor to another.

Assume that you have a 6-core chip that you want to use to solve a given problem:

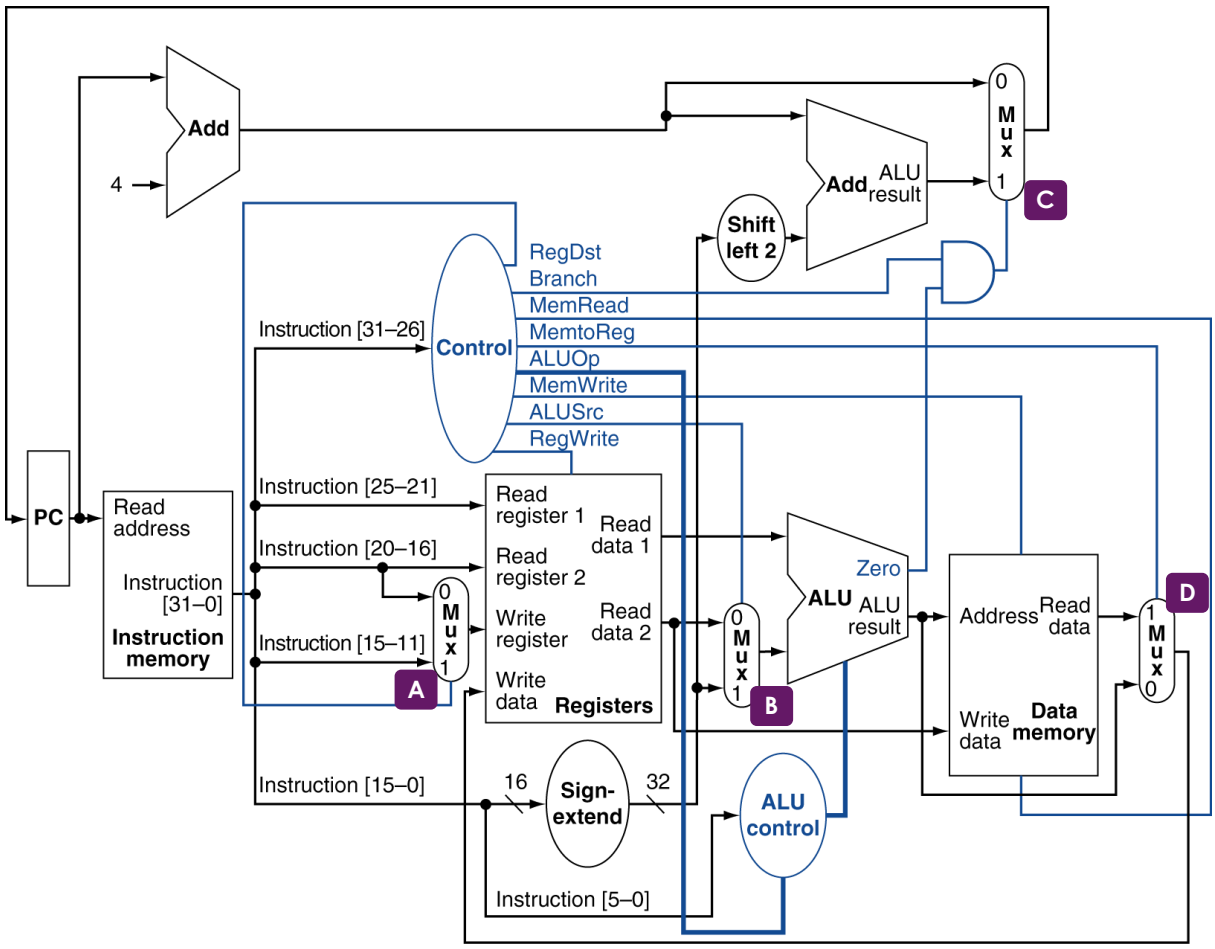
- You can use as few as 1 core or as many as 6 cores to solve the problem.
- The problem requires 450,000 iterations of a main loop to complete.
- Each loop iteration requires 100 clock cycles.
- Any startup overhead can be ignored (i.e. instructions outside of the loop to instantiate variables, etc.).

If more than 1 core is used to solve a problem, the overhead must be added to the total execution time. This overhead is a function of number of cores and can be defined as: $t_{overhead}=10(X-1)$ where X is the number of processors used and $t_{overhead}$ is the number of cycles spent in overhead code per iteration of the loop. For example total overhead for using 2 cores will be: 10*(2-1)*450000=450000 cycles.

A. How many cores should be used to solve this problem the fastest? (4p)
B. What is the speedup for it? (2p)

Task 3 – Datapath (8p)

The following diagram shows a simple datapath for MIPS instruction set. In this datapath there are 4 multiplexers (Mux), which are named A, B, C and D. For each of them, explain their role in the datapath with examples.



Task 4 – Pipelining (5p)

Consider the following sequence of MIPS instructions:

```
SW   R16, -100(R6)
LW   R4, 8(R16)
ADD  R5, R4, R4
```

a) Indicate dependencies and their type. (1p)
b) Assume there is no forwarding in this pipelined processor. Indicate hazards and add NOP instructions to eliminate them. (1p)
c) Assume there is full forwarding. Indicate hazards and add NOP instructions to eliminate them. (1p)
d) What is the total execution time of this instruction sequence without forwarding and with full forwarding? What is the speedup achieved by adding full forwarding to a pipeline that had no forwarding? (Assume that the latency for all the stages is equal and is 250ps for non-forwarding system and 300ps with full forwarding). (2p)

Task 5 – Cache Performance (6p)

For two direct-mapped cache designs with a 32-bit address, the following bits of the address are used to access the cache.

Cache	Tag	Index	Offset
A	31-10	9-5	4-0
B	31-12	11-6	5-0

a) What is the cache line size (in words) for caches A and B? (1p)
b) How many entries does each caches have? (1p)
c) What is the ratio between total bits required for such a cache implementation over the data storage bits? (1p)

Starting from power on, the following byte-addressed cache references are recorded:

Address	0	4	16	132	232	160	1024	30	140	3100	180	2180

d) How many blocks are replaced if Cache A is used? (2p)
e) What is the hit ratio for Cache A? (1p)

Task 6 – Multiprocessing (6p + 3p BONUS)

Serena Kogan is a programmer at Cyberdyne Systems, and she has been asked to parallelize an old program so that it runs faster on modern multicore processors.

A. She parallelizes the program and discovers that its speedup over the single-threaded version of the same program is significantly less than the number of processors. She finds that many cache invalidations are occurring in each core's data cache. What program behavior could be causing these invalidations? Explain. (1p)
B. She modifies the program to fix this first performance issue. However, now she finds that the program is slowed down by a global state update that must happen in only a single thread after every parallel computation. In particular, the program performs 90% of its work (measured as processor-seconds) in the parallel portion and 10% of its work in this serial portion. The parallel portion is perfectly parallelizable. What is the maximum speedup of the program if the multicore processor had an infinite number of cores? (3p)
C. How many processors would be required to attain a speedup of 4? (2p)

BONUS: The rest are extra parts of this task. <u>You do not have to do them</u>, unless you are hoping to get bonus points.

In order to execute the program with parallel and serial portions more efficiently, Cyberdyne decides to design a custom heterogeneous processor with following properties:

- *This processor will have one large core (which executes code more quickly but also takes greater die area on-chip) and multiple small cores (which execute code more slowly but also consume less area), all sharing one processor die.*
- *When the program is in its parallel portion, all of its threads execute **only** on small cores.*
- *When the program is in its serial portion, the one active thread executes on the large core.*
- *Performance (execution speed) of a core is proportional to the square root of its area.*
- *Assume that there are 16 units of die area available. A small core must take 1 unit of die area. The large core may take any number of units of die area n^2, where n is a positive integer.*
- *Assume that any area not used by the large core will be filled with small cores.*

D. How large should Serena make the large core for the fastest possible execution of the program? (2p)
E. *What would the same program's speedup be if all 16 units of die area were used to build a homogeneous system with 16 small cores, the serial portion ran on one of the small cores, and the parallel portion ran on all 16 small cores? (1p)*