CDT204 – Computer Architecture

Date: Oct. 30th 2017

Time: 8:10 – 13:30

Help: Small calculator is allowed in the exam

The exam has 35 points and 4 BONUS points. The grades will be awarded as follows:

- 3 : 18 Points
- 4 : 24 Points
- 5 : 30 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.
- Bonus questions will give you extra points (they raise the total over 36).
- In case you might have forgotten:

  $1G = 2^{10}M = 2^{20}K = 2^{30}$

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

Live long and prosper

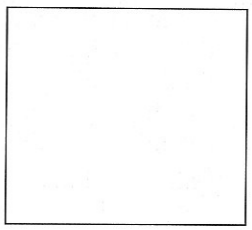
Task 1 – General (4p)

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

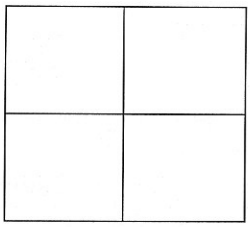
Task 2 – Performance (7p)

Consider the following three processors (X, Y, and Z) that are fabricated on a constant silicon area of 16A. Assume that the single-thread performance of a core increases with the square root of its area.

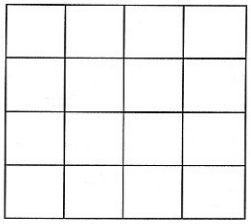
Processor X
1 large core of area 16A



Processor Y
4 medium cores of area 4A



Processor Z
16 small cores of area A



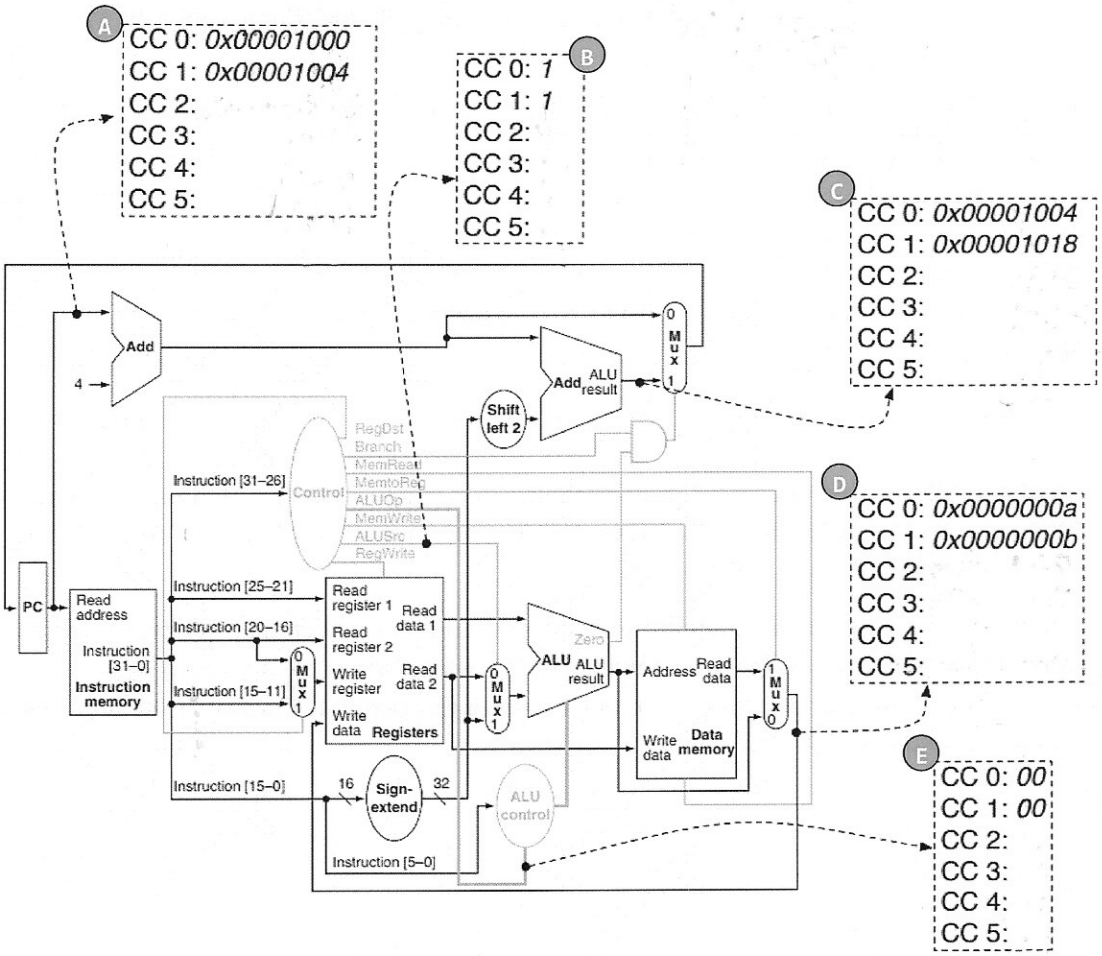
On each of the three processors, we will execute a workload where S fraction of its work is serial, and where 1-S fraction of its work is perfectly parallelizable. Assume that it takes time T to execute the entire workload using only one of the small cores of Processor Z.

A. As a function of S, plot the execution time of the workload on each of the three processors. (3p)
B. Which processor has the lowest execution time for the widest range of S? (max S) (1p)
C. Typically, for a realistic workload, the parallel fraction is not perfectly parallelizable. What are the three fundamental reasons why? (3p)

Task 3 – DataPath (5p)

For the processor below, fill in the blanks giving the values on the indicated wires for cycles 2-5 while running the following set of instructions. Cycles 0 and 1 are provided to get you started. You may use "U", if necessary, for undefined values. (5p)

```
TOP: lw  $t0, 0($s0)       # opcode=0x23, rs=0x10, rt=0x08, imm=0x0000
     lw  $t1, 4($s0)       # opcode=0x23, rs=0x10, rt=0x09, imm=0x0004
     beq $t0, $t1, LBL     # opcode=0x04, rs=0x08, rt=0x09, imm=0x0001
     sw  $t0, 4($s0)       # opcode=0x2b, rs=0x10, rt=0x08, imm=0x0004
LBL: addi $s0, $s0, 8      # opcode=0x08, rs=0x10, rt=0x10, imm=0x0008
     beq $zero, $zero TOP  # opcode=0x04, rs=0x00, rt=0x00, imm=0xfffb (-6)
```



Task 4 – Pipelining (5p)

In this question, assume that all branches are perfectly predicted (this eliminates all control hazards). Assume, we have only one memory (for both instructions and data), and there might be a structural hazard. To resolve this, the pipeline must be stalled in some cycles.

```
        SW  $s2, 0($s3)
        OR  $s1, $s2, $s3
        BEQ $s2, $s0, Label (Assume $s2== $s0)
        OR  $s2, $s2, $s0
Label : ADD $s1, $s4, $s3
```

A. Show the pipelined execution for each instruction. (draw a table, show stages, etc). (3p)
B. Identify and explain where the pipeline stalls. (2p)

Task 5 – Cache Performance (6p)

Assume a cache design with 32-bit addresses and byte-addressable memory, the following bits of the address are used to access the cache:

	Tag	Index	Offset
Bits	31-10	9-5	4-0

A. What is the cache line (block) size? (1p)
B. How many entries does the cache have? (1p)

Now assume the following code:

```
struct Student {
  unsigned Id;
  int Grades[15];
  char Name[128];
}
...
unsigned N = 100000;
Student* Students = new Student[N];
float getavg(unsigned id) {...}
```

The function `getavg()` gets a student Id as input and returns the average of his/her Grades. The array is not sorted by Ids. The cache is defined as above.

C. What is the number of cache misses while running `getavg()` function, assuming that the matching entry is placed at the position `i` in the Students array? (0<i<N) (2p)
D. How can you modify the program to reduce the number of cache misses during the search? How many cache misses do you have now? (2p)

Task 6 – Multiprocessing (4p + BONUS 4p)

Flynn's taxonomy in its classical form (represented in the table below) defines 4 different classes in computer architecture from a multiprocessing point of view.

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

A. Explain each of the classes and give examples for each of them. (2p)
B. How would you classify GPGPUs (i.e. NVIDIA Tesla) in terms of Flynn's taxonomy? How are GPGPU architectures different from vector architectures (i.e. Intel AVX)? (2p)

[The following is the BONUS part, do it if you have time]
Assume you have an array **A** of **n** integers with **n** in the order of tens of millions. You need a kernel that replaces the even values stored in **A** with 1 and odd numbers with 0. Look at the following implementation:

```
void foo (int* A, int n) {
  /*
  blockIdx.x is the blockId of the current thread block
  blockDim.x is the number of threads in a thread block
  threadIdx.x is the threadId within its thread block
  */
  int i = blockIdx.x*blockDim.x+threadIdx.x;
  if(i<n && A[i]%2==0)
        A[i] = 1;
  else
        A[i] = 0;
}
...
foo <<<n/256+1,256>>> (A, n); // kernel invocation
```

C. What kind of inefficiency can you detect in the above implementation? Explain which they are, referring to the GPU architecture seen in class, and provide a new more-efficient implementation which fixes them. (3p BONUS)
D. Where is most of the time spent running the two kernels? (1p BONUS)