**Fall 2016 Pretest : COMP 7300/06 Advanced Computer Architecture (100 pts)**

**Advice:**

**Sift through all the full test and work first on the easy or familiar exercises/questions.**

1. **If you cannot do a question/exercise, let me know the reasons (10% credit):**
   1. **I never saw this**
   2. **I saw this but I still do not understand it**
   3. **I can answer if I had some time to refresh**
   4. **Other reason**

**Fall 2016: COMP 7300 Advanced Computer Architecture**

**Pretest (100 pts)**

**Grading policy:**

¼ Credit for correct answer

¾ Credit for well written and solid **justification/facts/arguments**. Show your work, show the expressions you use, show the operations that yield your final result.

**For the final, unless otherwise stated in a specific question, we assume a CPU with a 32-bit address bus.**

**A) Introduction**

**Questions**

1. **Questions (Brief responses)**
2. **(2 points)** What does a RISC (architecture) stand for?

Reduced Instruction Set Computer

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 47 | 5 | 6 |

1. **(2 points)** How does CISC compare with RISC?

Complex Instruction Set Computer. CISC has more instructions that perform more complex/sophisticated operations. CISC instructions are of variable length (fixed for RISC). Because of simpler instructions RISC is easier to pipeline.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 42 | 5 | 11 |

**b) Exercise**

We consider a program ***P*** executed on an imaginary processor with a clock rate of 4 GHz. This table provides for each type of instruction the number of executions and the number of cycles. Answer the questions based on this table.

|  |  |  |
| --- | --- | --- |
|  | Number of instructions | (CPI) Clock cycle Per Instruction |
| Load | 250 | 3 |
| Store | 250 | 3 |
| Arithmetic | 200 | 2 |
| Logical | 100 | 1 |

1. **(2 points)** How many clock cycles in total are needed to execute Program ***P***?

Execution time = Sum(Number of Instrucions I \* Number of Clock Cycles for I)

= 250 \* 3 + 250 \* 3 + 200 \* 2 + 100 \* 1 = 2,000 clock cycles = 2,000 **cc**

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 51 | 4 | 2 |

1. **(2 points)** What is the average CPI (average number of clock cycles per instruction) for Problem ***P***?

CPI = total number of clock cycles/total number of cycles

= 2,000/(250+250+200+100) = 2,000 / 800 =2.5 cc/instruction

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 41 | 15 | 2 |

1. **(2 points)** What is the latency (in **ns**) of **ONE** *Load* instruction?

Latency = number of clock cycles \* (duration of a cc) = number of clock cycles \* (1/clock frequency)

= 3 \* 1/ 4GHz = 3 \* ¼ \* 10-9 =3 \* 0.25 ns = 0.75 ns

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 36 | 14 | 7 |

1. **(2 points)** What is the latency (in **ns**) of **ONE** *Logical* instruction?

Latency = number of clock cycles \* (duration of a cc) = number of clock cycles \* (1/clock frequency)

= 1 \* 1/ 4GHz = 1 \* ¼ \* 10-9 =1 \* 0.25 ns = 0.25 ns

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 34 | 14 | 10 |

1. **(2 points)** How long does Program ***P*** take to execute (in **microseconds**)?

Execution time = total number of clock cycles \* duration of one clock cycle = 2,000 \* 0.25 ns

= 500 ns = 0.5 s.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 33 | 15 | 10 |

**B) Hierarchical Memory**

**a) (3 points) Justification of a hierarchical memory**

So far, static RAM (SRAM) cost about $2,000 per GB while magnetic disks cost about $1.0 per GB for an access time of about 10 ms. A company just announced a discovery of a new material that allows to build a non volatile memory for $0.20 per GB with an access time less than one ns. If this claim is true, how many memory levels would next generation computers have?

Since the new non volatile memory 1) is faster than SRAM (currently used for caches) and 2) is cheaper secondary storage (magnetic disks), than memory for next generation would have only one level built using the new material. .

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 24 | 24 | 10 |

**b) (3 points) New computing paradigm**

A new computing paradigm resulted in a new programming language that produces programs whose 90% of instructions are branches (of which very few are loops) and using only binary trees as data structures. Would caching work better for this new programming language (than for current programming languages you know/use)?

First, 90% of branches (with few loops) will produce poor temporal locality

Second, references to binary trees will result in poor special locality. Therefore, caching would work worse.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 27 | 14 | 17 |

**c) Cache Optimization**

Computer architects settled on using some associativity (2-way or 4-way) for the caches.

1. **(2 points)** What is a direct-mapped cache?

In a direct mapped cache, an item stored at address A in the main memory will be always cached at the same memory location () in the cache.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 19 | 25 | 14 |

1. **(3 points)** How does a direct-mapped cache differ from an associative one?

Contrary to a direct-mapped cache, an item will be always stored at the same location in a cache.

In an n-way associative cache, an item may be stored at n potential locations in a cache.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 15 | 21 | 22 |

**c) Cache Basics**

We consider a CPU with a 10-bit address bus, a 2-way 256 byte cache with a 16 byte block size.

1. **(4 points)** How wide (number of bits) is the offset?

Since the block size is 16 bytes = 24, then offset is 4 bits

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 21 | 17 | 20 |

1. **(4 points)** How many sets are there in the cache?

The total number of blocks in a cache = Cache size / block size

= 256 bytes / 16 byte per block = 16 blocks. The cache is 2-way associative,

then each set contains 2 blocks, then this cache will contain 16/2 = 8 sets.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 22 | 15 | 21 |

1. **(4 points)** How wide is the index?

Since there are 8 (23) sets, then the index is 3 bits wide.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 17 | 14 | 27 |

1. **(3 points)** How wide is the tag?

An address consists of a tag, an index, and an offset.

|  |  |  |
| --- | --- | --- |
| Tag | Index | Offset |

Therefore the tag size = address size – index size – offset size

= 10 bits – 3 bits – 4 bits = 3 bits

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 15 | 15 | 27 |

1. **(3 points)** Provide the tag, index, and offset for Address 545 (0x221)

|  |  |  |
| --- | --- | --- |
| Tag (3 bits) | Index (3 bits) | Offset (4 bits) |

545 = 0x221 = 10 0010 0001 = 100 010 0001

|  |  |  |
| --- | --- | --- |
| Tag (3 bits) | Index (3 bits) | Offset (4 bits) |
| 100 | 010 | 0001 |

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 15 | 6 | 37 |

**d) Assembly**

1) **(6 points)** Write in assembly (using any microprocessor you know) a program that 1) adds two 32 bits numbers A and B stored at physical addresses 0x1200 and 0x1204, respectively and 2) stores the sum at address 0xA200.

If using MIPS assembly language for example, we would have : (note that register $0 always contains value 0). We ignore the carry.

LW $1,$1200($0) Register $1 🡨 MEM[$0 + 0x1200]

LW $2,$1204($0) Register $2 🡨 MEM[$0 + 0x1204]

ADD $1,$1,$2 Register $1 🡨 $1 + $2

SW $1, 0xA200($0) MEM[$0 + 0xA000] 🡨 $1

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 35 | 14 | 9 |

2) Consider the following program ***P***. Note than **[**0x1000**]** refers to **content of address** 0x1000 while #0x1000 refers to an **immediate** value. AX is a 16 bit register.

MOV AX, 0x0800 AX 🡨 value 0x800

ADD AX, 0x0800 AX 🡨 AX + value 0x800

MOV [0x1000], AX [0x1000]🡨 AX

1. **(3 points)** What are the contents of AX and [0x1000], respectively at the end of the execution?

AX will contain 0x0800 + 0x0800 = 0x1000. [0x1000] will contain the value 0x1000.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 26 | 21 | 11 |

1. **(6 points)** How many memory references does Program P initiate in **TOTAL**?

MOV AX, 0x0800 One **instruction** memory reference

ADD AX, 0x0800 One **instruction** memory reference

MOV [0x1000], AX One **data** memory reference and One **instruction** memory reference

In total, this program makes 4 memory references.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 4 | 35 | 19 |

**e) Virtual memory**

We consider a 32 bit address bus and a 256 KB physical memory. Page size is 32 KB.

1. **(3 points)** Why do we use a virtual memory on modern computers?

In short, virtual memory is used to run programs concurrently even if the sum of their sizes exceeds the main memory size.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 33 | 17 | 8 |

1. **(9 points)** Assuming a validity bit, a dirty bit, and **two** reference bits, what is the size of the page table? **Show the different steps to determine the page table size.**

Page Table Size = number of entries \* the size of one entry

Number of entries = Number of pages in the virtual space = Virtual Space Size / Page size

= 232 / 32 210 = 232 / 215 = 217.

Each entry contains a validity bit (1), a dirty bit (1), two reference bits (2) and frame number

We need the number of bits in a frame number. The physical space contains

256 KB. The number of frames = Total size / size of one frame = 256 KB / 32 KB

= 28 KB / 25 KB = 23 = 8 frames

Therefore the size of a page table has validity bit (1), a dirty bit (1), two reference bits (2) and 3 bits for the frame number.

Page Table size = 217 \* (1 + 1 +2 + 3) = 7 217 bits.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 5 | 13 | 40 |

1. **(3 points)** Suppose that Entry ***n*** in the page table contains the value floor(n/2^11). What is the physical address associated with virtual address (0x10002400)?

Page number = Address / Page Size = 0x10002400 / 32 K = 0x10002400 / 215 = 0x10002400 >> 15

= (1 0000 0000 0000 0)2

The page table entry pg indicated by the page number contains n/2^11 = n >> 11, then the frame number

is equal to (1 0000 0000 0000 0)2 >> 11 = (1 00)2

The offset of the virtual address is equal to virtual address % page size = (0x10002400) % 32K = 0x2400

Concatenate frame number and offset will give a physical address = 10 0010 400 = 0x22400

**Another method:**

0x10002400 = 0001 0000 0000 0000 0010 0100 0000 0000

Offset = the 15 (Page size = 32 K = 215) least significant bits of the physical address = 010 0100 0000 0000

Page # = Virtual address with the offset taken off the virtual address = 0001 0000 0000 0000 0

Frame # = content of the entry = page number >> 11 = 0001 00

Physical address = concatenation of frame # and offset = 0001 0 0010 0100 0000 0000 = 0x22400

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 2 | 13 | 43 |

**C) Instruction Level Parallelism**

**a) Pipeline Performance**

This exercise is about deriving the speed up for an unbalanced pipeline.

Consider a monocycle CPU with **4** (**FOUR**) "operations". We assume that all instructions need the 4 operations. Operations take 200 ps, 400 ps, 200 ps, and 200 ps, respectively.

1. **(3 points)** What is the latency of one instruction on the monocycle CPU?

Latency of one instruction = sum of all stages/operations

= 200 ps + 400ps + 200 ps + 200 ps = 1,000 ps = 1 ns

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 40 | 7 | 11 |

1. **(3 points)** What should be the clock frequency for the monocycle CPU?

The period of the frequency must be equal to the latency of one instruction.

Frequency = 1/period = 1/1ns = 1/10-9s = 109 Hz = 1 GHz.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 29 | 11 | 18 |

1. **(3 points)** What is the throughput (bandwidth) of the monocycle CPU?

This CPU delivers one instruction every ns, then

the throughput = number of instructions executed in one second

= 1 instruction/10-9s = 109 instruction /second

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 17 | 20 | 21 |

1. **(3 points)** Consider a program P with n instructions. What is the execution time for Program P (expression as a function of n)?

Each instruction takes one ns, then ***n*** instructions will take ***n*** ns

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 29 | 10 | 19 |

5) We want to pipeline the CPU above with one stage per operation. We neglect the buffer time between stages. All the questions below apply to the pipeline .

a) **(3 points)** What is the latency for one instruction for the pipelined CPU?

The latency will not change, the CPU must still perform all operation for each instruction.

Therefore, the latency of one instruction = sum of all stages/operations

= 200 ps + 400ps + 200 ps + 200 ps = 1,000 ps = 1 ns

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 6 | 26 | 26 |

b) **(3 points)** What should be the clock frequency for the pipelined CPU?

The frequency of the clock must be set such that the period is at least the latency of the slowest operation. The slowest operation takes 400ps, then the frequency = 1/400ps = 1/0.4 ns = 2.5 GHz.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 15 | 13 | 30 |

c) **(3 points)** What is the throughput (bandwidth) of the pipelined CPU?

Since the CPU is pipelined, it will execute one instruction every 400 ps,

the throughput = number of instructions executed in one second

= 1 instruction/400 10-12s = 2.5 109 instruction per second.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 15 | 16 | 27 |

1. **(3 points)** Consider a program P with n instruction. What is the execution time for Program P? (Make sure to take into account the phase to fill up the pipeline and wind it up)

n instructions will take the time to fill up the pipeline (3 cc) and n cc to to produce the n instructions. Execution time = (n + 3) cc = (n + 3) \* 400 ps = 0.4 (n +3) ns

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 10 | 18 | 30 |

1. **(3 points)** What is the expression of the speed up (monocycle versus pipelined)? If n tends to infinity, what is the speed up?

Speed up = Time of Slowest Execution/ Time of fastest Execution = n/0.4(n+3)

When n tends to infinity, speed up will reach 1/0.4=2.5

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 8 | 13 | 37 |

**D) Data-Level Parallelism**

a) **(3 bonus points)** Briefly describe a *vector architecture*

Briefly, this is a SIMD machine: single instruction multiple data. The same instruction is applied simultaneously to multiple data.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 17 | 7 | 34 |

b) **(3 bonus points)** Using Flynn classification/taxonomy, how do you classify a vector architecture?

See slides Set1—CAQA5e\_ch1 slides : Slide # 7

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 11 | 5 | 42 |

c) **(3 bonus points)** What is the KEY gain/improvement of a vector architecture?

This architecture minimizes the memory accesses due for instructions fetching.

|  |  |  |
| --- | --- | --- |
| Don’t Know | Know, but Wrong | Right |
| 15 | 9 | 34 |