CENG3420 Homework 3

Due: Apr. 08, 2018

Please submit PDF or WORD document directly onto blackboard.
DO NOT SUBMIT COMPRESSED ZIP or TARBALL.

Solutions

Q1 (**10%**) In this exercise, we will look at different ways cache capacity affects overall performance. In general, cache access time is proportional to cache capacity. Assume that main memory accesses take 68 ns. The following Table 1 data for L1 caches attached to each of two processors, P1 and P2.

Table 1: Question 1

	L1 Size	L1 Miss Rate	L1 Hit Time
P1	2 KB	13.4%	0.72 ns
P2	4 KB	7.8%	0.87 ns

1. Assuming that the L1 hit time determines the cycle time for P1 and P2, what are their respective clock rates? (5%)
2. What is the AMAT (Average Memory Access Time) for P1 and P2? (5%)

A1

1. $\frac{1}{0.72} \approx 1.39$ GHz, $\frac{1}{0.87} \approx 1.15$ GHz
2. $0.72 + 13.4\% \times 68 = 9.832$ s, $0.87 + 7.8\% \times 68 = 6.174$ s
   or: $0.72 \times (1 - 13.4\%) + 68 \times 13.4\% = 9.73552ns$, $(1 - 7.8\%) \times 0.87ns + 7.8\% \times 68 = 6.10614ns$

Q2 (**15%**) What are differences between interrupt and DMA? (5%) Figure 1 shows the connection among CPU, DMA control and Peripheral. Please describe the process when data is transmitted from peripheral to memory. (10%)

A2

1. Interrupt: implemented by programming, I/O transmission at low speed, deal with complex issue, response requirement until end current instruction; DMA: implemented by hardware, simple I/O transmission, response requirement until end bus cycle.
2. (a) Peripheral sends "DREQ" signal to DMA controller;
   (b) DMA controller sends bus requirement signal "HRQ" to CPU;
   (c) CPU sends bus response signal "HLDA" to DMA controller and DMA controller controls bus.
   (d) DMA controller sends DMA response signal "DACK" to Peripheral to tell Peripheral that DMA controller has controlled bus and the transmission of data will be allowed;

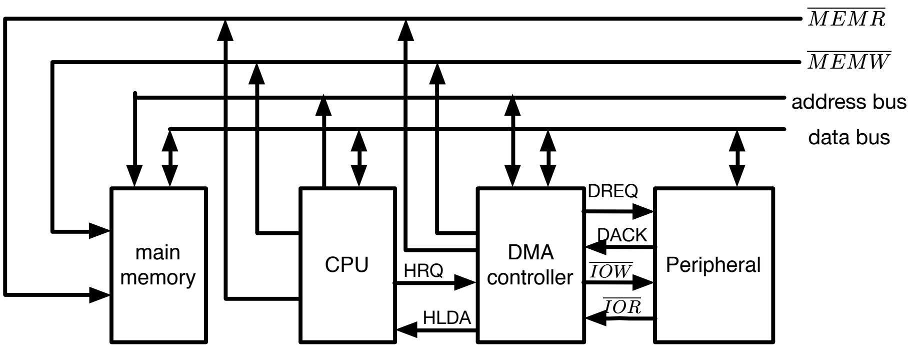

Figure 1: the connection among CPU, DMA controller and Peripheral

(e) According to main memory counter, DMA controller sends address signal as main memory address. Meanwhile, the counter of main memory address increases 1;

(f) DMA controller sends $\overline{IOR}$ signal to Peripheral to read data to bus. Meanwhile, it sends "$\overline{MEMW}$" signal and the data of data bus is written to main memory unite chosen by address bus;

(g) The transmission counter will decreases 1;

(h) Repeat from (e) to (g) until transmission counter is decreased to 0. And the signal "HRQ" becomes low level and CPU controls bus again.

Q3 (**15%**) Consider the following portions of two different programs running at the same time on five processors in a symmetric multi-core processor (SMP). Assume that before this code is run, both x, y and z are 2.

```
Core 1:  x = x + 1;
Core 2:  y = z + 1;
Core 3:  w = x - y;
Core 4:  z = x + 1;
Core 5:  r = w + z;
```

1. What are all the possible resulting values of w, x, y, z and r? For each possible outcome, explain how we might arrive at those values. You will need to examine all possible interleavings of instructions. (10%)
2. How could you make the execution more deterministic so that only one set of values is possible? (5%)

A3 1. {x=3, y=3 or 4 or 5 or, z=3 or 4, w=0 or -1 or -2 or -3 or 1, r=2 or 1 or 0 or -1 or 3 or 4 or 5} all possible combinations

2. We could set synchronization instructions after each operation so that all cores see the same value on all nodes.

Q4 (**10%**) Consider matrix multiplication $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$, where $\mathbf{C} \in \mathbb{R}^{m \times n}$, $\mathbf{A} \in \mathbb{R}^{m \times k}$ and $\mathbf{B} \in \mathbb{R}^{k \times n}$. You are given the following code to perform $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$

```
for (int i = 0; i < m; ++i) {
    for (int j = 0; j < n; ++j) {
        C[i][j] = 0;
        for (int t = 0; t < k; ++t) {
            C[i][j] += A[i][k] * B[k][j];
        }
    }
}
```

Clearly the code takes $\mathcal{O}(mnk)$ time. We would like to improve the actual running time. Please give a strategy by parallel computation without race.

A4

```
parallel for (int i = 0; i < m; ++i) {
        parallel for (int j = 0; j < n; ++j) {
                C[i][j] = 0;
                for (int t = 0; t < k; ++t) {
                        C[i][j] += A[i][k] * B[k][j];
                }
        }
}
```

Q5 (**10%**) We will upgrade a processor for CSE department. For performing application and programming, the computation speed of the new processor is 10X faster than the old processor. Assume that the old processor spends 40% time to compute and 60% to wait for I/O response. Please compute speedup.

A5

$$\text{speedup} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

Q6 (**15%**) In a system, the main memory size is 4 MB, the virtual memory size is 1 GB. What is the bit width of the virtual address and physical address, respectively? Assume that the page size is 4 KB, what is the page length?

A6

1. main memory: 22 bit width
2. virtual memory: 30 bit width
3. page length: $\frac{1GB}{4KB}$

Q7 (**15%**) Given an original code as follows:

```
Loop:
    L.D      F0,0 (R1)
    ADD.D    F4, F0, F2
    S.D      F4, 0 (R1)
    DADDUI   R1, R1, #-8
    BNE      R1, R2, Loop
```

1. Please revise the original code to the code with loop unrolling.(10%)
2. Based on the revised the code with loop unrolling, please revise the code with pipeline scheduling.(5%)

A7

```
    Loop:
     L.D      F0, 0 (R1)
     ADD.D    F4, F0, F2
     S.D      F4, 0 (R1); drop DADUI & BNE
     L.D      F6, -8 (R1)
     ADD.D    F8, F6, F2
     S.D      F8, -8 (R1); drop DADDUI & BNE
     L.D      F10, -16 (R1)
     ADD.D    F12, F10, F2
     S.D      F12, -16 (R1); drop DADDUI & BNE
     L.D      F14, -24 (R1)
     ADD.D    F16, F14, F2
     S.D      F16, -24 (R1)
     DADDUI   R1, R1, #-32
     BNE      R1, R2, Loop
```

```
Loop:
    L.D      F0, 0 (R1)
    L.D      F6, -8 (R1)
    L.D      F10, -16 (R1)
    L.D      F14, -24 (R1)
    ADD.D    F4, F0, F2
    ADD.D    F8, F6, F2
    ADD.D    F12, F10, F2
    ADD.D    F16, F14, F2
    S.D      F4, 0 (R1)
    S.D      F8, -8 (R1)
    DADDUI   R1, R1, #-32
    S.D      F12, 16 (R1)
    BNE      R1, R2, Loop
    S.D      F16, 8 (R1); 8-32 = -24
```

or

Table 2: Solution 7

	ALU or branch	Data transfer	CC
Loop:	L.D F0,0(R1)		1
	L.D F6,-8(R1)		2
	L.D F10,-16(R1)	ADD.D F4, F0, F2	3
	L.D F14,-24(R1)	ADD.D F8, F6, F2	4
	S.D F4, 0(R1)	ADD.D F12, F10, F2	5
	S.D F8, -8(R1)	ADD.D F16, F14, F2	6
	S.D F12, -16(R1)		7
	DADDUI R1, R1, #-32		8
	S.D F16, 8(R1)		9
	BNE R1, R2, Loop		10

Q8 (**10%**) We will examine space/time optimizations for page tables. Table 3 shows parameters of a virtual memory system.

Table 3: Question 8

	Virtual Address (bits)	Physical DRAM Installed	Page Size	PTE Size (byte)
a	43	16 GB	4 KB	4
b	38	8 GB	16 KB	4

1. For a single-level page table, how many page table entries (PTEs) are needed? How much physical memory is needed for storing the page table? (5%)
2. Using a multilevel page table can reduce the physical memory consumption of page tables, by only keeping active PTEs in physical memory. How many levels of page tables will be needed in this case? And how many memory references are needed for address translation if missing in TLB? (5%)

A8

1. a:
   virtual address 43 bits, physical memory 16 GB,
   page size 4 KB or 2^{15} bits, page table entry 4 bytes or 2^5 bits,
   #PTE=43-15=28 bits or 2^{18} K entries, PT physical memory = 2^{18} K $\times 4\text{bytes}$ = 2^{20} KB;
   b:
   virtual address 38 bits, physical memory 8 GB,
   page size 16 KB or 2^{17} bits, page table entry 4 bytes or 2^5 bits,
   #PTE = 38-17=19 bits or 2^9 K entries,
   PT physical memory $= 2^9 K \times 4\text{bytes} = 2^{11}$ KB.
2. a:
   4 KB page/4 bytes PTE = 2^{10} pages indexed per page. Hence with 2^{28} PTEs will need $\text{ceil}(28/10) =$ 3-level page table setup. Each address translation will require at least 3 physical memory accesses;
   b:
   16 KB page/4 bytes PTE = 2^{12} pages indexed per page. Hence with 2^{19} PTEs will need $\text{ceil}(19/12) =$ 2-level page table setup. Each address translation will require at least 2 physical memory accesses;