Introduction to Computer Systems
2014 Fall Midterm Examination

Name_____ Student No._____ Score_____

Problem 1: (10 points)

1. A: LRU

B: Prefetching

C: Write-back

2.

Problem 2: (7 points)

1. 012102:

200212:

210012:

201022:

3. [1] [2] [3]

Problem 3: (9 points)

1.

2.

Problem 4: (8 points)

1

2

Problem 5: (8 points)

1.

2

3

4

Problem 6: (28 points)

1. [1] [2] [3] [4]

2. [1] [2] [3] [4] [5]
   [6] [7] [8] [9] [10]
   [11] [12] [13] [14] [15]
   [16] [17] [18] [19] [20]

3. [1] [2] [3]
   [4] [5] [6]
   [7] [8] [9]
   [10] [11] [12]

4. [1] [2] [3] [4]

Problem 7: (6 points)

1. [1] [2]

2. [1] [2] [3]

3. [1] [2]

4.

Problem 1: Warm up (10 points)

1. What kind of locality do the following technologies explore?
   NOTE: temporal or spacial locality? (2'x3)

   A: LRU
   B: Prefetching
   C: Write-back

2. How can we block a signal explicitly? (2') How about ignoring a signal explicitly? NOTE: Please write down the function calls you will use (2')

Problem 2: Process (7 points)

```
1  #include "csapp.h"
2  int main(void) {
3      int iter = 3, i;
4      for (i = 0; i < iter; i++) {
5          int ret = !fork();
6          if((ret + i) % 2 == 0) {
7              printf("%d", ret);
8          } else {
9              printf("2");
10             break;
11         }
12     }
13 }
```

1. Suppose **printf** would immediately output the result. Please determine whether the following outputs are possible or not. (4')

   A. 012102
   B. 200212
   C. 210012
   D. 201022

2. If we comment the **break** in **line 10**, how many "0", "1" and "2" will be printed on the screen? (3')

   0: __[1]___ 1:__ [2]___ 2:__[3]___

Problem 3: Signal (9 points)

```
1  #include "csapp.h"
2  pid_t pid;
3  int n = 100;
4  void handler1(int sig) {
5      printf("receive signal from child\n");
6  }
7  int main(void) {
8      int status;
9      Signal(SIGCHLD, handler1);
10     if ((pid = Fork()) == 0) {
11         int i;
12         for(i = 0; i < n; i++) {
13             Kill(getppid(), SIGCHLD);
14         }
15     } else {
16         while(waitpid(-1, &status, 0) > 0);
17     }
18
19     exit(1);
20 }
```

1. Is the output of this program deterministic or non-deterministic (2')? Why? (3')

2. Suppose some guy sends **SIGKILL** from console to the child process when it is running, then the child process will exit and a **SIGCHLD** signal will be sent to the parent process. How can the parent process tell the difference between this signal and the previous ones? You need to modify **handler1** to achieve this? (4')

Problem 4: CPU (8points)

```
int a[100000], i;
for (i = 1; i < 10000; ++i) { /* some code from A, B or C */ }
```

```
A: printf("%d\n", a[i]);
B: a[i] = i * (i/43 >> 2) * (i + 23 + 1024%i);
C: a[i] = i;
```

1. Which inner loop code (A, B or C) can benefit from the store buffer most and Why? (2'+2')
2. If CPU finds that a branch is predicted incorrectly, it will reset some states of its components. Which of the following components will NOT be affected and Why? (2'+2')
   A. Store Buffer B. FP Add Units
   C. Register File

Problem 5: Modern Processor (17points)

```
1  .L2:
2  mulsd a(,%rax,8), %xmm0  // a is the address of an array.
3  movsd %xmm0, a(,%rax,8)
4  addq  $1, %rax
5  cmpq  %rdx, %rax
6  jl .L2
```

Assume that there is only ONE double-precision multiplication unit in the processor. All other CPU resources are UNLIMITED. The latency and issue time of the units are given in the below table. We don't take cache into consideration, thus all memory operations have the identical latency. NOTE: Two independent load/store operations will not affect each other. But if there is any dependence, the second operation will wait for the first one to complete.

operation	Integer		Double-precision	
	latency	Issue	latency	issue
Addition	1	1	2	1
Multiplication	3	1	5	1
Load/Store	3	1	3	1

1. Please draw the data-flow graph of above code like Figure 5.13 in ICS book. (4')
2. Please draw the data-flow graph like Figure 5.15 in ICS book and bold the critical path in your graph. (3'+2')

3. Please calculate the CPE on current CPU. If we have UNLIMITED number of multiplication units, how much is CPE? (2'+2')
4. Now we swap the instruction at line 3 and line 4, please give out the CPE with original LIMITED number of multiplication units and explain your answer. (2'+2')

Problem 6: Cache (28 points)

Consider a 12-bit machine with a two-way L1 cache, the size of each cache line is 4 bytes and the size of L1 cache is 32 bytes.

1. please fill the following blanks (4')

   How many sets does the L1 cache has: [1]

Field	Length(bit)
Tag	[2]
Set	[3]
Offset	[4]

Besides the L1 cache, we also have a two-way L2 cache, the contents of the L2 cache are as follows, with Hex notation. Suppose the L1 cache is empty, and the memory access is 1-byte words. Both L1 and L2 cache use LRU replacement policy. (NOTE: L1 cache miss will find data in L2 cache and the data will also maintain in L2 cache.)

Set	Tag	Valid	Byte0	Byte1	Byte2	Byte3	Tag	Valid	Byte0	Byte1	Byte2	Byte3
0	0x01	1	0x12	0x44	0x66	0x55	0x00	0	--	--	--	--
1	0xEB	0	--	--	--	--	0xCC	0	--	--	--	--
2	0x06	0	--	--	--	--	0x0B	1	0x10	0x20	0x30	0x40
3	0x77	1	0xDE	0xFF	0x99	0xFE	0xE7	1	0x11	0x22	0x33	0x44
4	0x00	1	0x12	0x34	0x56	0x78	0x60	0	0x78	0x56	0x34	0x12
5	0xCC	0	--	--	--	--	0x38	0	--	--	--	--
6	0x56	0	--	--	--	--	0x99	0	--	--	--	--
7	0x78	1	0xDD	0xAC	0xAA	0x11	0x77	1	0xAB	0xCD	0x1E	0x13

2. We have several sequentially memory access, please fill the following blanks. NOTE: if unknown fill in "--", if L1 is hit, then L2 "hit or not" attribute should fill "YES". (3'x4)

Order	Address	L1 Set	L1 Hit or not (Yes/No)	L2 Set	L2 Hit or not(Yes/NO)	Byte Returned
1	0x013	0	No	4	Yes	0x78
2	0xC13	[1]	[2]	[3]	[4]	[5]
3	0xEFF	[6]	[7]	[8]	[9]	[10]
4	0xE10	[11]	[12]	[13]	[14]	[15]
5	0x010	[16]	[17]	[18]	[19]	[20]

3. Suppose both L1 and L2 cache change to full-associative cache and the contents of L2 cache are as follows. The L1 cache is also empty as question2, and the memory access is 1-byte word. Both L1 and L2 cache use LRU replacement policy.

Set	Tag	Valid	Byte0	Byte1	Byte2	Byte3	Tag	Valid	Byte0	Byte1	Byte2	Byte3
0	0xEB3	0	--	--	--	--	0x000	0	--	--	--	--
	0x010	1	0x12	0x44	0x66	0x55	0x004	1	0x12	0x34	0x56	0x78
	0x062	0	--	--	--	--	0x0B0	1	0x10	0x20	0x30	0x40
	0x774	1	0xDE	0xFF	0x99	0xFE	0xE71	1	0x11	0x22	0x33	0x44
	0x304	0	0xDD	0xAC	0xAA	0x11	0x3BF	1	0xAB	0xCD	0x1E	0x13
	0xCCa	0	--	--	--	--	0x382	0	--	--	--	--
	0x56f	0	--	--	--	--	0x993	0	--	--	--	--
	0x601	0	0x78	0x56	0x34	0x12	0xCCf	0	--	--	--	--

Now we have several sequentially memory access, please fill the following blanks (NOTE: You do not need to consider the influence of the memory access in question 2) (2'x4)

Order	Address	L1 Hit or not (Yes/No)	L2 Hit or not(Yes/NO)	Byte Returned
1	0x013	No	Yes	0x78
2	0xC13	[1]	[2]	[3]
3	0xEFF	[4]	[5]	[6]
4	0xE10	[7]	[8]	[9]
5	0x010	[10]	[11]	[12]

4. TRUE or FALSE (4')

[1]. Higher associativity decreases the vulnerability of the cache to thrashing due to conflict misses.
[2]. Higher associativity is hard to make fast
[3]. Larger cache will increase the hit rate
[4]. Larger cache will decrease the hit time.

Problem 7: Optimization (21points)

Assume that there is a single LRU cache with a 16 bytes block size. "**A**" is an **X*Y*Z** array of int (the size of int is 4 bytes). The array size **Y** and **Z** are so large that a single matrix row does not fit in the cache. All loop indexes (**i**, **j** and **k**) are stored in registers, which not require any load or store instructions. "**sum**" does not interfere the access to "**A**" and "**B**". The access to "A" and the access to "B" do not interfere each other. The memory alignment and compiler driven optimizations are NOT considered.

```
//abs is from cmath.h, and
//return absolute value
#define X 20
#define Y 40
#define Z 60

int A[X][Y][Z];
int B[Y];
```

```
int func(int* sum){
  int i, j, k;
  *sum = 0;
  for (k = 0; k < Z; k++)
    for (j = 0; j < Y; j++)
      for (i = 0; i < X; i++)
        *sum += A[i][j][k] + B[j];
  for(i=0; abs(*sum) < 511037; i++)
    *sum += i;
  return 1;
}
```

1. In above code, what is the expected cache miss rate of the access to array A and B? (2'x2) A: ___[1]___ B: ___[2]___

2. Which order of the following loops exhibits the best locality? (4')
   NOTE: the order of above example is **k** on the outside, **j** in the middle, and **i** on the inside.

   Outside: ___[1]___ Middle: ___[2]___ Inside: ___[3]___

3. What is the expected cache miss rate of the access to A and B in the order of loops exhibiting the best locality? (2'x2)

   A: ___[1]___ B: ___[2]___

4. Please further optimize the function **func()** based on question 2. (9')

   *Hint: 1) You need to write down the code of the optimized **func()**, using annotation to explain what techniques you use; 2) Use at least 3 techniques; 3) The "**sum**" is usually far less than 511037 at first.*