Name:________________ Login: cs61c-___

ANSWERS

CS61C, Spring 2011

Final Examination

Please read the following CAREFULLY:

- The use of electronic devices is not permitted on this exam.
- One (1) page of notes, front and back, is permitted on this exam, not counting the MIPS green card, which will be provided for you.
- You will have 3 hours (180 minutes) to complete this exam.
- Looking at another student's exam, talking about the exam during the exam period, or talking about the exam after the exam period but before all CS61C students have taken the exam, is prohibited.
- Violations of the above will be academic dishonesty. Violators will be dealt with harshly. You do not want to find out what "harshly" means. Please don't make us inform you.
- You should write your name and login on every page of this exam, so that we may associate detached pages with your exam.

Please read, fill out, and sign the following:

I, ____________________________, have read the preceding points, and understand them fully.

Name: ____________________

SID: ____________________

Login: cs61c-________

TA/Section: ____________________

Signature: ____________________

Problem (pts)	Points Received
1 (10)	
2 (10)	
3 (10)	
4 (8)	
5 (8)	
6 (16)	
7 (14)	
8 (14)	
Total (90)	

Name:________________ Login: cs61c-___

1. True or False

Directions: Circle the correct answer.

a. (True [✓ False]) Linear strong scaling is easier to obtain than linear weak scaling as the number of processors increases.

b. ([✓ True] False) An algorithm that performs 2 floating point operations on each element of a 100000000 integer array will most likely show performance that is limited by memory bandwidth on one of our lab computers.

c. (True [✓ False]) Cache blocking increases peak performance by increasing the amount of data reuse in cache.

d. (True [✓ False]) As the problem size increases, thread creation overhead becomes an ever-greater problem if an algorithm uses a constant number of threads.

e. (True [✓ False]) You notice that your algorithm obtains better speedup when you switch from double to single precision arithmetic than from the addition of SSE intrinsics. This implies that the algorithm is bound by the machine's floating point capability.

f. (True [✓ False]) To ensure correctness, a compiler must take into account dynamic multiple instruction issue.

g. ([✓ True] False) On MIPS, executing an lw instruction once can require two page faults (assuming proper alignment).

h. (True [✓ False]) On MIPS, modifying $s0 in a function without saving the caller's value will cause an exception.

i. (True [✓ False]) A MIPS exception handler follows the same calling convention as ordinary MIPS functions.

j. ([✓ True] False) The compulsory cache miss rate can be reduced by changing cache parameters (e.g. associativity, block size, cache size).

Name:________________ Login: cs61c-___

2. OpenMP can set you free....

Consider the following OpenMP snippet:

```
int values[100000];
#pragma omp parallel
{
        int i = omp_get_thread_num();
        int n = omp_get_num_threads();
        for (; i < 100000; i += n) {
                values[i] = i;
                #pragma omp barrier
        }
}
```

#pragma omp barrier is a synchronization construct that causes a thread reaching it to continue execution only after all other threads have reached the barrier.

Suppose each core has a single level of non-shared, write-back, write-allocate, direct-mapped 32 KB data cache with 64-byte cache blocks. Assume data cache accesses other than those to the 'values' array are negligible and that all data caches are initially empty. Assume ints are 32 bits. Assume there are as many cores as threads.

a. If the snippet is run with one thread, how many data cache misses for the values array will there be?

6250 (2 pts)

b. If the snippet is run with two threads (each allocated a separate core), what is the maximum number of data cache misses for the values array?

100 000 (3pts; 1pt for 2x (a))

c. In five words or less, name the phenomenon that the difference or lack of difference between your answer to (a) and (b) illustrate.

"false sharing" or "ping ponging" or "cache invalidations" (2 pts)

d. Using just two threads, if we remove the barrier, *could* the number of data cache misses for accesses to the values array decrease by more than a factor of 2 from your answer in (b)? Explain briefly.

Yes. (1pt) Because one thread could get ahead of ~~each~~ the other, avoiding the ping ponging effect. (2pts)

Name:__________________ Login: cs61c-____

3. To stall, or not to stall....

Suppose to add support for some complex arithmetic instructions, a 5-stage MIPS pipeline is modified into a 6-stage MIPS pipeline by splitting the execute stage into two execute stages. Assume in this *6-stage pipeline*, the result of the ALU is not available until the second execute stage, but the ALU takes its inputs in the first execute stage. (Assume the ALU is internally pipelined so this extra stage does not introduce any structural hazards.)

Assume we implement *all* forwarding paths to avoid hazards. For each the following assembly snippets (a-f), determine whether a stall is needed between the two instructions with a standard 5-stage MIPS pipeline *and* with this hypothetical 6-stage pipeline. (Circle *all* that apply.)

a. addu \$t0, \$t1, \$t2
subu \$t1, \$t2, \$t3

1 five-stage pipeline stalls six-stage pipeline stalls [✓ neither stalls]

b. addu \$t0, \$t1, \$t2
subu \$t3, \$t0, \$t4

1 five-stage pipeline stalls [✓ six-stage pipeline stalls] neither stalls

c. lw \$t0, 0(\$t1)
sw \$t0, 0(\$t2)

1 five-stage pipeline stalls six-stage pipeline stalls [✓ neither stalls]

d. addu \$t0, \$t1, \$t2
sw \$t0, 0(\$t3)

1 five-stage pipeline stalls six-stage pipeline stalls [✓ neither stalls]

e. lw \$t0, 0(\$t1)
addu \$t2, \$t0, \$t1

2 [✓ five-stage pipeline stalls] [✓ six-stage pipeline stalls] neither stalls

f. addu \$t0, \$t1, \$t2
beq \$t3, \$zero, Label

1 five-stage pipeline stalls six-stage pipeline stalls [✓ neither stalls]

7

Name:__________________ Login: cs61c-___

g. Write a C expression to determine when a 5-stage MIPS pipeline must stall between a 'lw' and a following 'addiu' instruction. Use “L.rs”, “L.rt”, “L.immediate”, and so on to represent the fields of the lw instruction and “A.rs”, “A.rt”, and so on to represent the fields of the addiu instruction. You do not need to compare the opcodes.

```
L.rt == A.rs && A.rs != 0
```

3 pts.

-1 for ~~omit~~ omitting zero check

Name:__________________ Login: cs61c-___

4. Reality Check

To make things simple, let's assume that the data word is just 4 bits wide.

a. What is the minimum number of extra redundant bits that you need to *detect* a single bit error for a 4-bit data word?

2 pts

i. None
[✓ ii. 1]
iii. 2
iv. 3
v. 4

b. What is the minimum number of extra redundant bits that you need to *correct* a single bit error for a 4-bit data word?

2 pts

i. None
ii. 1
iii. 2
[✓ iv. 3]
v. 4

c. The following bit pattern includes Hamming SEC/DED (single error correction, double error detection) using even parity to calculate the correcting/detecting code.

Bit number:	1 2 3 4 5 6 7 8
Data:	1 1 0 0 1 1 1 0

1 pt

i. ([✓ YES] NO) Does it have a bit error?

ii. If so, which bit has the error? Leftmost bit is 1, rightmost bit is 8.

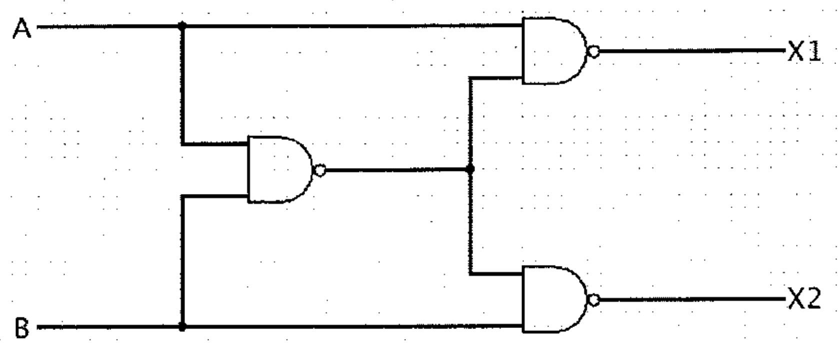
3pts

- 1
- 2
- 3
- 4
- 5
- 6
- [✓ 7]
- 8

Name:________________ Login: cs61c-___

5. NAND, NAND, NAND, ...

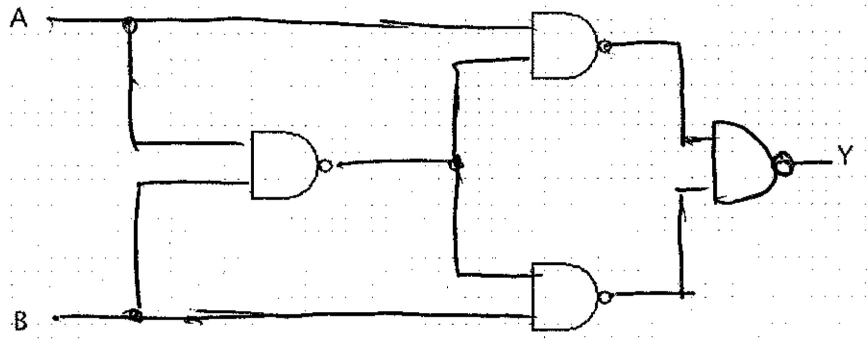
a. Fill in the truth table for the following circuit:



3pts

A	B	X1	X2
0	0	1	1
0	1	1	0
1	0	0	1
1	1	1	1

b. Draw in another 2-input NAND gate and wire it up to make a circuit which satisfies the following truth table: (Note: Your solution should only have four 2-input NAND gates, otherwise you get no points.)



3pts

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

c. Which of the following gates has the identical functionality of the above circuit? Circle one.

NAND NOR AND OR [✓ XOR] XNOR

2pts

Name:________________ Login: cs61c-___

6. Revenge of the AMAT

- Suppose that for 1000 memory references, we have
  - 40 misses in direct-mapped L1$ (i.e., the miss rate is 4%)
  - 20 misses in 2-way set associative L1$ (i.e., the miss rate is 2%)
  - 10 misses in L2$ (i.e., the global miss rate is 1%)
- Further,
  - L1$ hits in 1 cycle
  - L2$ hits in 10 cycles
  - Miss to main memory costs 100 cycles
- Assume that we have 1.5 memory references per instruction (i.e., 50% loads and stores). In other words, for 1000 instructions we have 1500 memory references.
- Ideal CPI is 1.0 (if we had a 100% hit rate in L1$)

a. What is the local miss rate for L2$...

i. assuming a direct-mapped L1$?

$$10/40 = 25\%$$

ii. assuming 2-way set associative L1$?

$$10/20 = 50\%$$

b. What is the AMAT (Average Memory Access Time)

i. assuming a direct-mapped L1$?

$$\text{HIT TIME}_{L1} + \text{MISS RATE}_{L1} \times (\text{HIT TIME}_{L2} + MR_{L2} \times MP_{L2})$$
$$= 1 + 4\%(10 + 25\% \times 100) = 1 + 4\% \times 35 = 1 + 1.4 = \boxed{2.4}$$

ii. assuming 2-way set associative L1$?

$$= 1 + 2\%(10 + 50\% \times 100) = 1 + 2\% \times 60 = 1 + 1.2 = \boxed{2.2}$$

c. How much faster is the AMAT for a 2-way set associative cache? Give your answer as a ratio.

$$\frac{2.4}{2.2} = 1.091$$

Name:_______________ Login: cs61c-___

d. What is the average number of memory stall clock cycles per **reference**
   i. assuming a direct-mapped L1$?

EITHER $\frac{40}{1000} \times (10 + 10/40 \cdot 100) = 4\% \times 35 = \boxed{1.4}$

OR $AMAT_{DM} - 1 = 2.4 - 1 = \boxed{1.4}$

   ii. assuming 2-way set associative L1$?

EITHER $\frac{20}{1000} \times (10 + 10/20 \cdot 100) = 2\% \times 60 = \boxed{1.2}$

OR $AMAT_{2WAY} - 1 = 2.2 - 1 = \boxed{1.2}$

e. What is the average number of memory stall clock cycles per **instruction**
   i. assuming a direct-mapped L1$?

$1.5 \times 1.4 = \boxed{2.1}$

   ii. assuming 2-way set associative L1$?

$1.5 \times 1.2 = \boxed{1.8}$

f. How much faster would a program run using a 2-way set associative cache?

$$\frac{1+2.1}{1+1.8} = \frac{3.1}{2.8} = 1.107$$

g. Are the answers for AMAT (6c) and program execution time (6f) above the same? Explain why or why not.

NO. INSTRUCTIONS EXECUTE 1.5 MEMORY REFERENCES ON AVERAGE, SO THE DIFFERENCE IS LARGER + IT IS ADDED TO CPI IDEAL

Name:________________ Login: cs61c-___

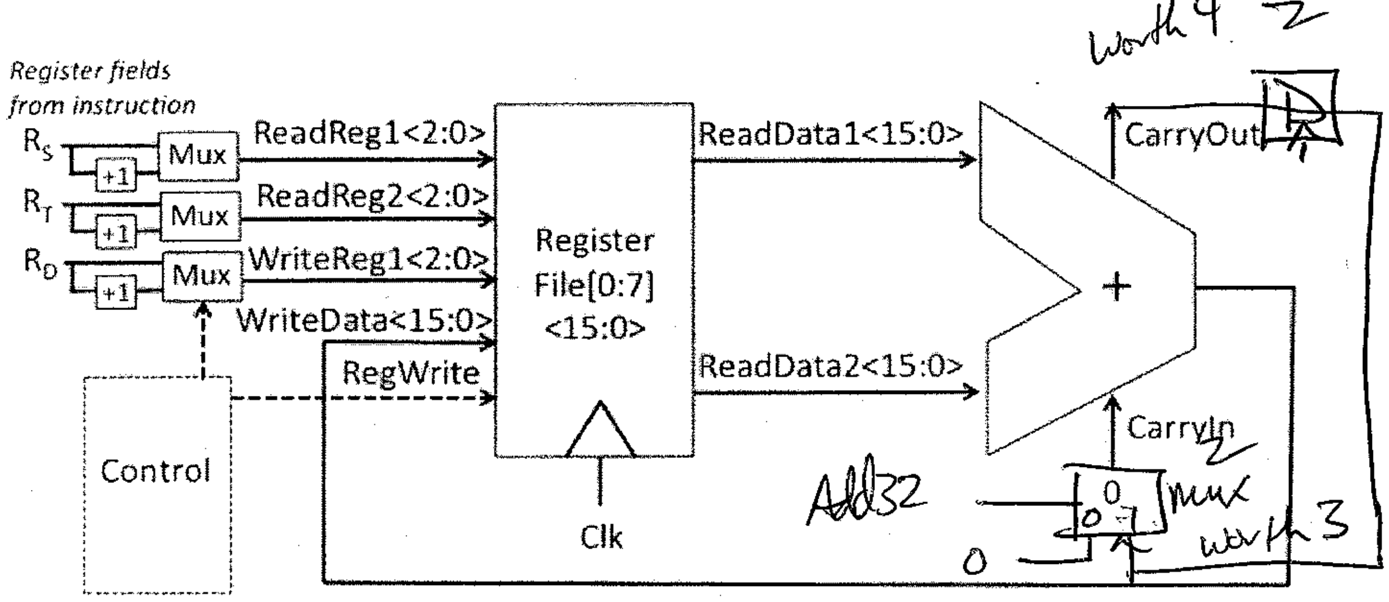
7. Analyze This

Your task is to determine how to support an instruction set with 32-bit wide operands using a 16-bit wide datapath, similar to the one you designed for Project #4.

The datapath consists of a register file organized as 8 x 16-bit registers (numbered 0 to 7) and a 16-bit wide ALU that implements the standard addition and subtraction operations. Like your project, register 0 is not special, and can be written as well as read. You need only be concerned with implementing the Register-to-Register operations.

To form 32-bit wide operands, registers are paired. Two registers in an even-odd pair (e.g., 0-1, 2-3, 4-5, 6-7) can store a full 32-bit quantity.

The following 16-bit wide datapath reads two registers, forms their sum (or other arithmetic operation), and writes back a 16-bit result in one clock cycle:



a. By extending the datapath to perform a 32-bit operation over *MORE THAN ONE clock cycle*, modify the datapath above with AS FEW PIECES OF ADDITIONAL HARDWARE as possible to implement 32-bit computations. Show your changes on the drawing above.

7 pts ↓

3-4 pts for blind effort.

$A_{31-16} + B_{31-16} \rightarrow$ carry in D

$A_{15\,0} + B_{15\,0}$ carry in D

-3 for correctly doing 3-cycle also.

Name:________________ Login: cs61c-___

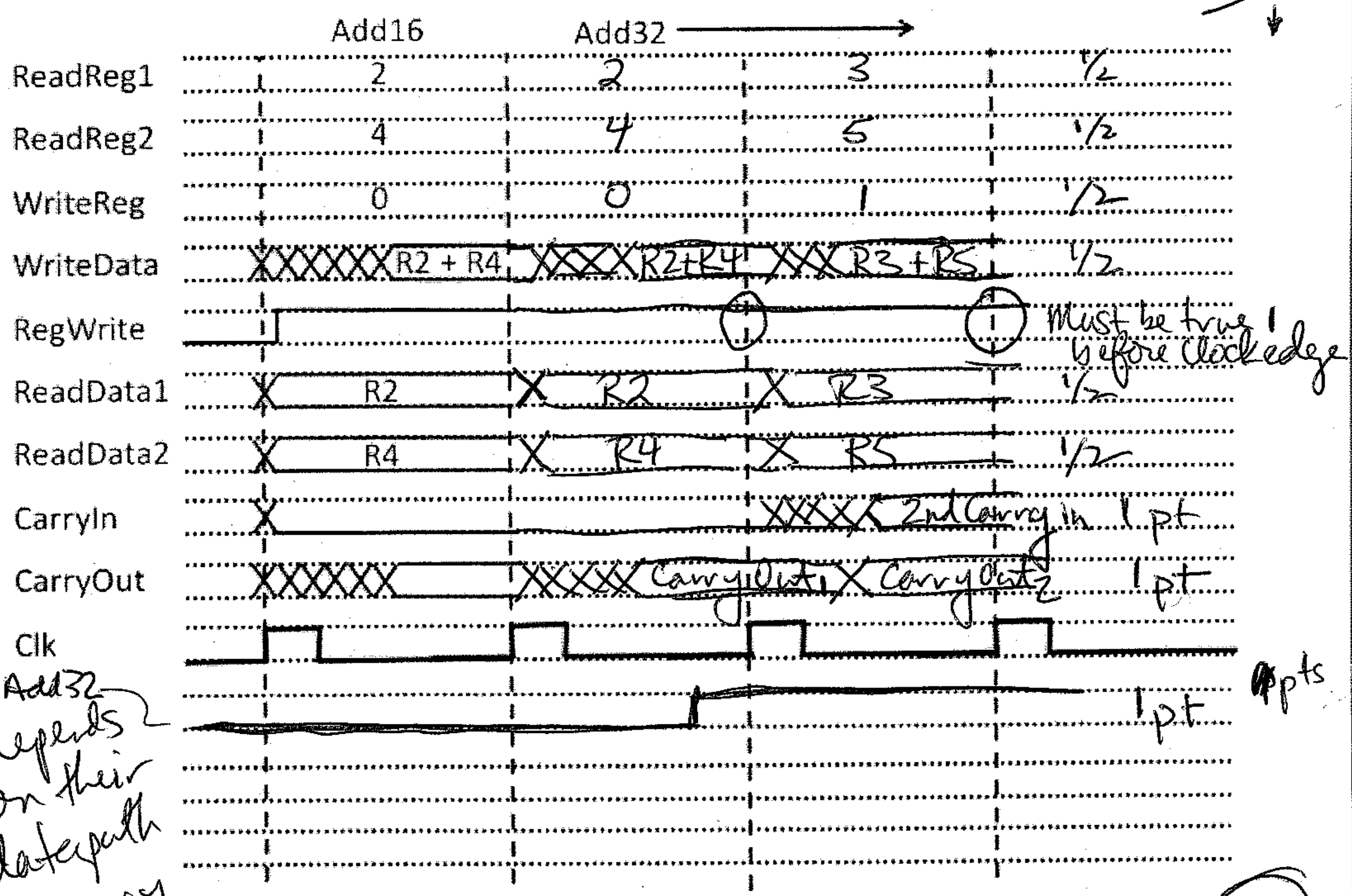
b. The first cycle of the following timing diagram shows a 16-bit register-to-register addition for the instruction
ADD16 0, 2, 4 (which performs the operation R0 <- R2 + R4) .

The next several cycles represent the execution of a 32-bit version of the instruction ADD32 0, 2, 4 (which performs the operation (R1, R0) <- (R3, R2) + (R5, R4)).

Note that the CarryOut signal is unknown at the start of the cycle (it will have some value of course). It becomes a final 0 or 1 value some delay after the sum of the contents of R2 and R4 emerge from the adder.

Extend the timing waveforms for the Add32 instruction and add timing waveforms for any additional signals you need to control the datapath additions you included above:

1 pt each



9 pts

depends on their datapath be wary of clock edges

−1 for swapping high-order & low order

7pts

Name:________________ Login: cs61c-___

8. One, two, three... SIMD!

a. SIMDize the following code:

```
void count( int n, float *c ) {
         for( int i = 0; i < n; i++ )
       c[i] = i;
}
```

Enter your solution by filling in the spaces provided. Assume n is a multiple of 4. (_mm_set1_ps(x) returns a __m128 with all four elements set to x.)

4 pts

-3 for getting this wrong

```
void countfast( int n, float *c ) {

    float m[4] = { __0__, __1__, __2__, __3__ };
    __m128 iterate = _mm_loadu_ps( m );

    for( int i = 0; i < ____n/4____; i++ ) {

        _mm_storeu_ps( ____c+i*4____, iterate );

        iterate = _mm_add_ps( iterate, _mm_set1_ps( ___4___ ) );
    }
}
```

b. Horner's rule is an efficient way to find the value of polynomial $p(x)=c_0x^{n-1}+c_1x^{n-2}+...+c_{n-2}x+c_{n-1}$:

```
float poly( int n, float *c, float x ) {
    float p = 0;
    for( int i = 0; i < n; i++ )
        p = p*x + c[i];
    return p;
}
```

Complete the following SIMD solution by filling in the blanks. Assume n is a multiple of 4.

```
float fastpoly( int n, float *c, float x ) {
    __m128 p = _mm_setzero_ps( );
    for( int i = 0; i < n; i += 4 ) {

4 pts       p = _mm_mul_ps( p, _mm_set1_ps( ___X*X * X *X___ ) );

1 pt        p = _mm_add_ps( p, _mm_loadu_ps( ___c+i___ ) );
    }

4 pts   float m[4] = { _X*X*X_, _X *X_, __X__, __1__ };
    p = _mm_mul_ps( p, _mm_loadu_ps( m ) );
    _mm_storeu_ps( m, p );

1 pt    return ___m[0]+ m[1] + m[2] + m[3]___;
}
```