Name ______________________________

Computer Architecture
EE 4720
Final Examination

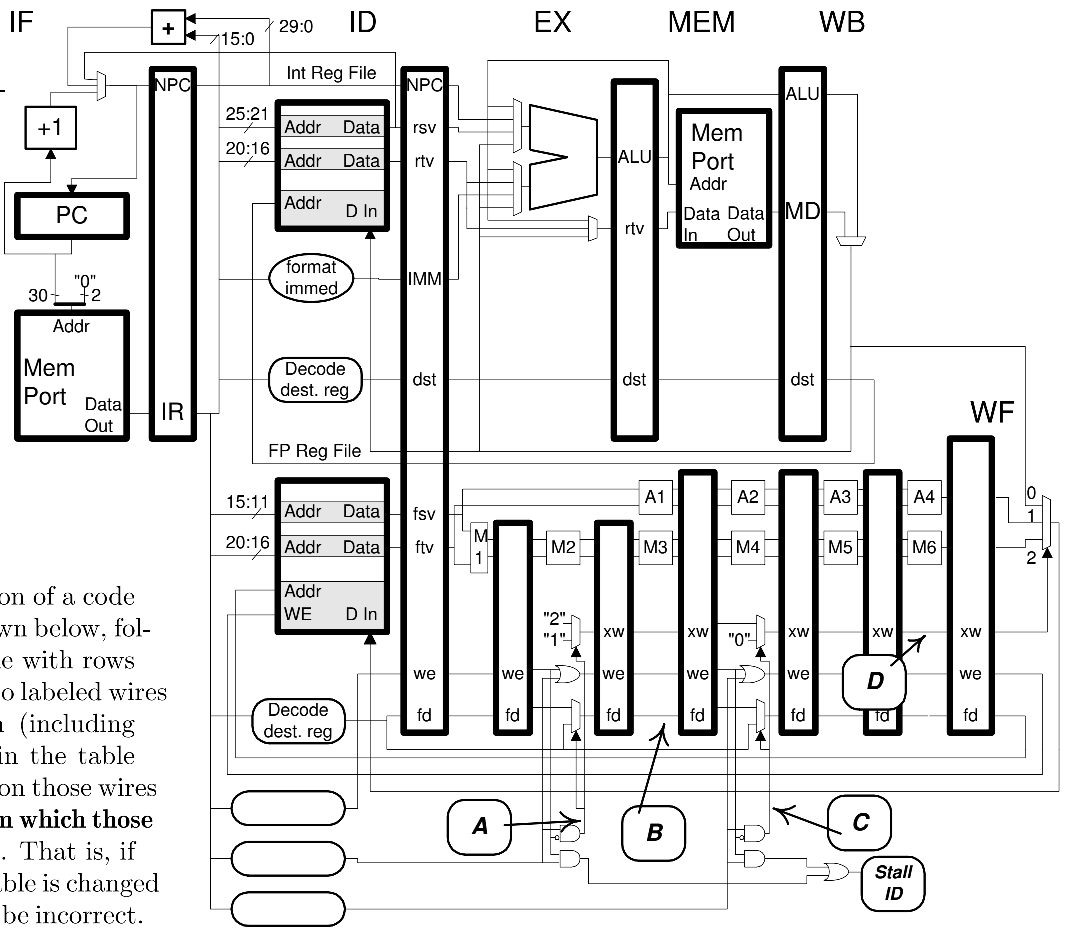
10 May 2007, 7:30–9:30 CDT

Problem 1	________	(20 pts)
Problem 2	________	(20 pts)
Problem 3	________	(20 pts)
Problem 4	________	(20 pts)
Problem 5	________	(20 pts)
Exam Total	________	(100 pts)

Alias ______________________________

Good Luck!

Problem 1: (20 pts) The statically scheduled MIPS implementation illustrated to the right is taken from the class notes. To avoid making things too easy some descriptions were removed from logic blocks in the lower-left corner.



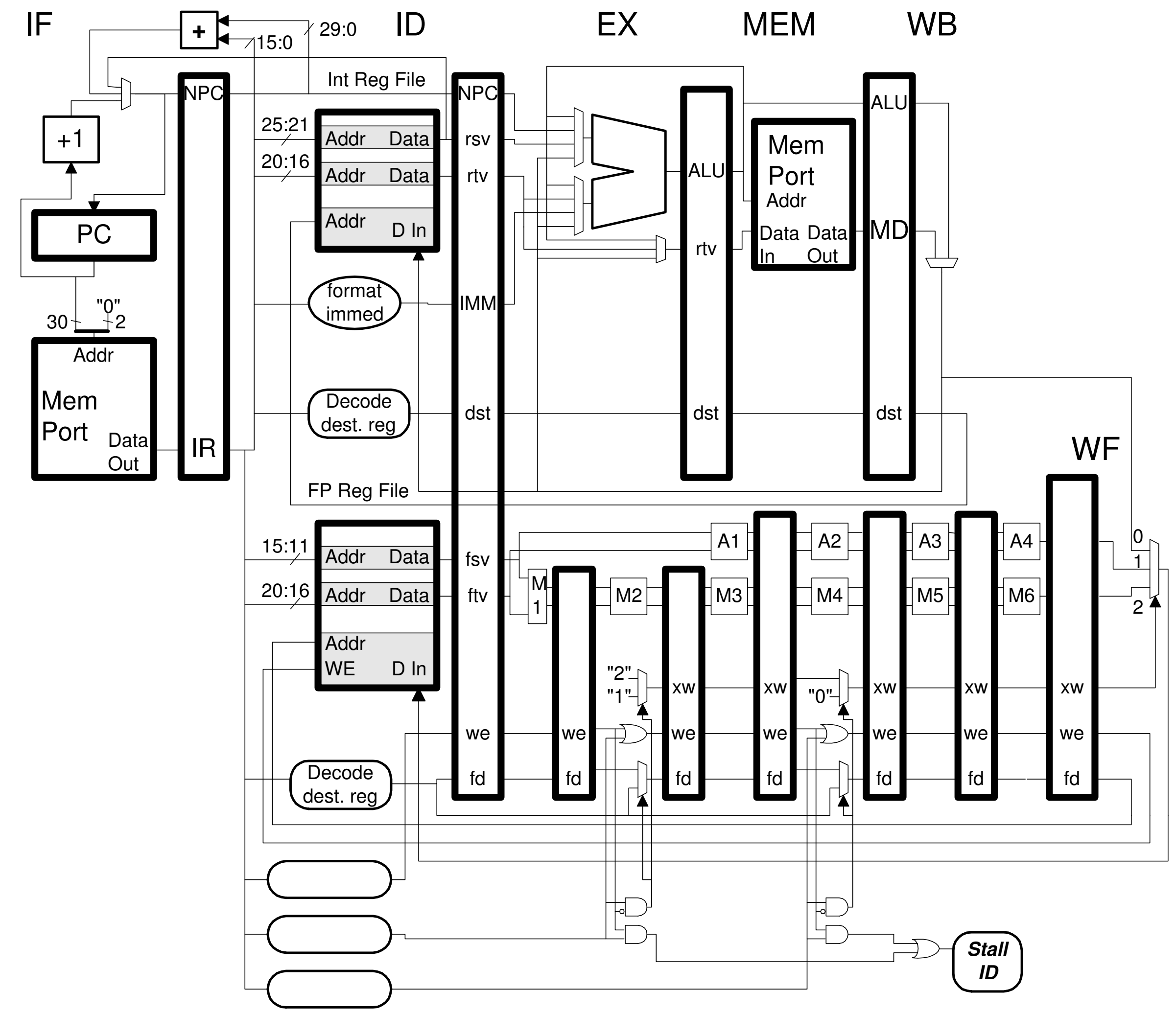
(*a*) The execution of a code fragment is shown below, followed by a table with rows corresponding to labeled wires in the diagram (including Stall ID). Fill in the table showing values on those wires **only for cycles in which those values are used**. That is, if a value in the table is changed execution must be incorrect.

☐ Complete the table, omitting unused values.

```
# Cycle                0     1     2     3     4     5     6     7     8     9     10
 mul.d f2, f12, f18    IF    ID    M1    M2    M3    M4    M5    M6    WF
 add.d f8, f10, f16          IF    ID    A1    A2    A3    A4    WF
 sub.d f6, f20, f14                IF    ID    ->    A1    A2    A3    A4    WF
 lwc1 f4, 0(r1)                          IF    ->    ID    ------->    EX    ME    WF
# Cycle                0     1     2     3     4     5     6     7     8     9     10
```

A:	
B:	
C:	
D:	
Stall ID:	

```
# Cycle                0     1     2     3     4     5     6     7     8     9     10
```



(*b*) Show the execution of the code below on the implementation assuming that all needed bypass are present. Don't forget to check for dependencies. (Instruction `mtc1` moves a value from an integer register to a floating-point register.)

☐ Pipeline diagram.

```
mtc1 f2, r7

add.s f3, f4, f2

add.s f6, f3, f8
```

(*c*) Add the bypass paths needed by the code above and show the control logic for the added paths. **Do not add unneeded bypass paths.** *Hint: Control logic should consist of two one-bit signals.*

☐ Bypass paths for code above.

☐ Control logic for added bypass paths.

Problem 2: (20 pts) Illustrated is the execution of some code on our dynamically scheduled MIPS implementation along with the contents of the ID register map, the commit register map, and the physical register file. The implementation itself is shown on the next page.

```
 # Cycle           0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
 mul.d f2, f4, f6   IF   ID   Q    RR   M1   M2   M3   M4   WF   C
 add.d f4, f2, f10       IF   ID   Q                   RR   A1   A2   A3   WF   C
 ldc1 f2,0(r1)                IF   ID   Q    EA   ME   WF                       C
 sub.d f2, f2, f8                  IF   ID   Q    RR   A1   A2   A3   WF                  C
 # Cycle           0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
ID Register Map
 F2:      12             9         71   99
 F4:      18                  51
 # Cycle           0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
Commit Register Map
F2:       12                                                 9                   71   99
F4:       18                                                                51
 # Cycle           0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
Physical Register File
  9                      [                                  2.1                 ]
 12          2.0                                                 ]
 18          4.0                                                                ]
 51                           [                                        4.1
 71                                [              2.2                                ]
 99                                     [                        2.3
 # Cycle           0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
```

(*a*) Answer the following

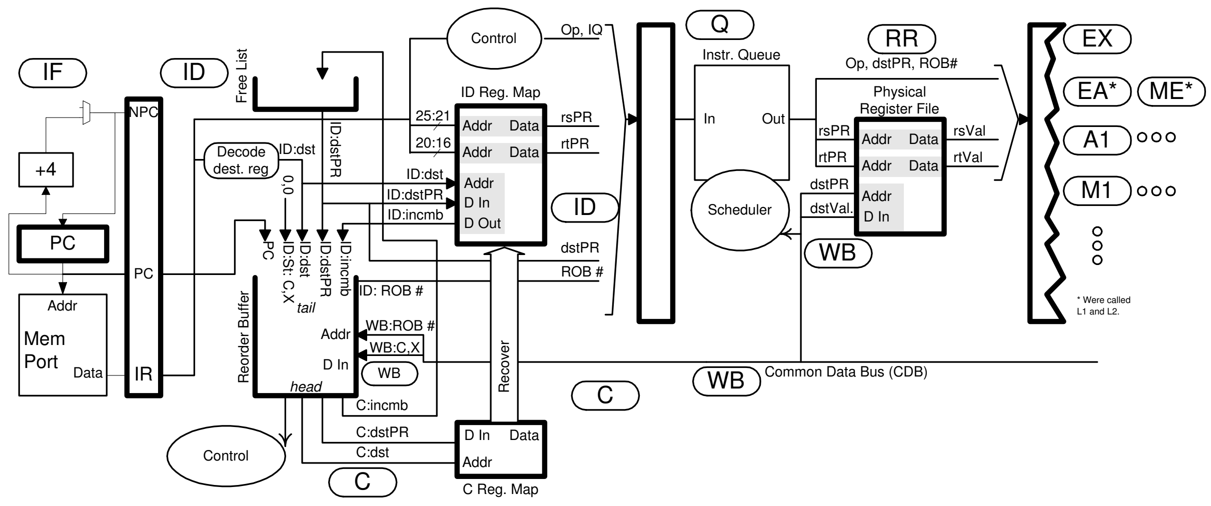
☐ Which physical register was allocated for `f4` in `add.d`?

☐ If one used the ID map to determine the value of `f2` in cycle 11, what value would one obtain? *Hint: It's a two-step process.*

☐ If one used the commit map to determine the value of `f2` in cycle 11, what value would one obtain? *Hint: It's a two-step process.*

☐ In cycle 11 where is the value of `f2` from `ldc1` located?

Problem 2, continued: Dynamically scheduled processor shown for reference.



Problem 2, continued:

```
 # Cycle            0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
 mul.d f2, f4, f6   IF   ID   Q    RR   M1   M2   M3   M4   WF   C
 add.d f4, f2, f10       IF   ID   Q                   RR   A1   A2   A3   WF   C
 ldc1 f2,0(r1)                IF   ID   Q    EA   ME   WF                            C
 sub.d f2, f2, f8                  IF   ID   Q    RR   A1   A2   A3   WF                  C
 # Cycle            0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
ID Register Map
 F2:      12             9         71   99
 F4:      18                  51
 # Cycle            0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
Commit Register Map
 F2:      12                                                     9                   71   99
 F4:      18                                                                    51
 # Cycle            0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
Physical Register File
  9                      [                                  2.1                      ]
 12          2.0                                                 ]
 18          4.0                                                                ]
 51                           [                                            4.1
 71                                [                   2.2                                     ]
 99                                     [                             2.3
 # Cycle            0    1    2    3    4    5    6    7    8    9    10   11   12   13   14
```

(*b*) Suppose that in cycle 11 the contents of physical register number 71 was somehow changed, perhaps due to a one-time problem. If the code executes as shown then the program runs correctly. But if a hardware interrupt happens (is taken) at the wrong time execution would be incorrect because of this change. Explain why, illustrate timing details on the diagram.

☐ Reason for incorrect execution.

☐ Timing details on diagram.

Problem 3: (20 pts) The MIPS code below runs on a system using a bimodal branch predictor of the indicated sizes. Branch outcomes are shown for each branch, the outcome patterns will continue to repeat.

```
BIGLOOP:
B1: 0x1000 beq r1, r2    T  T  T  T  T  N  T  N  N  T  T  T  T  T  T  N  T  N  N  T ...
... nonbranch insn.
...
B2: 0x1100 bne r3, r4     N  N  N  N  N  N  N  N  N  N  N  N  N  N  N  N  N  N  N  N  ...
nop
j BIGLOOP
nop
```

(*a*) What is the accuracy after warmup of a bimodal branch predictor with a 2^{14}-entry BHT on branch B1?

☐ 2^{14}-entry BHT bimodal accuracy on B1.

(*b*) What is the accuracy after warmup of a bimodal branch predictor with a 2^{4}-entry BHT on branch B1?

☐ 2^{4}-entry BHT bimodal accuracy on B1.

(*c*) What is the smallest BHT size for which one can obtain the same accuracy on branch B1 as a 2^{14} entry table? Explain.

☐ Smallest size for 2^{14} entry accuracy.

☐ Reason.

(*d*) Normally the BHT in a MIPS implementation is indexed starting at bit position 2 (omitting the 2 least-significant digits) of the branch PC (address). For the following questions think about answers to the preceding parts but answer the question for ordinary programs, not the code sample above.

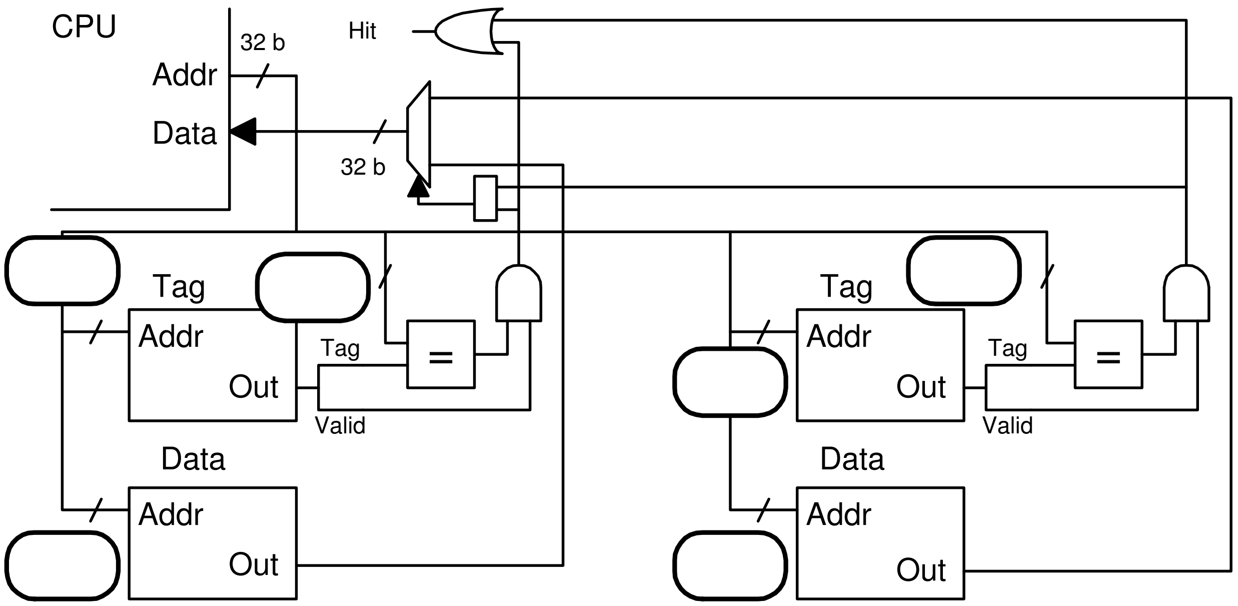
☐ Why might starting at position 3 or 4 be better?

☐ Why might starting at position 10 or 11 be worse?

Problem 4: (20 pts) The diagram below is for a 4-MiB (2^{22}-character) set-associative cache with a line size of 16 characters on a system with the usual 8-bit characters.

(*a*) Answer the following, formulæ are fine as long as they consist of grade-time constants.

☐ Fill in the blanks in the diagram.



☐ Show the address bit categorization. Label the sections appropriately. (Alignment, Index, Offset, Tag.)

Address:

☐ Associativity:

☐ Memory Needed to Implement (Indicate Unit!!):

☐ Show the bit categorization for a 64-way set-associative cache with the same capacity and line size.

Address:

Problem 4, continued:

(*b*) The code below runs on the same cache as the first part of this problem. Initially the cache is empty; consider only accesses to the array.

☐ What is the hit ratio running the code below? Explain

```
double sum = 0.0;
short *a = 0x2000000;    // sizeof(short) = 2 characters.
int i;
int ILIMIT = 1 << 10;    // = 2^10

for(i=0; i<ILIMIT; i++) sum += a[ i ];
```

(*c*) The code below runs on a **direct mapped cache** with the same line size and capacity as the cache from the first part. Initially the cache is empty; consider only accesses to the arrays. Choose `b`, `ILIMIT`, and `ISTRIDE` so that the cache is completely filled in the minimum number of iterations (minimum `ILIMIT`). (Every access should be a miss.)

☐ `b`, `ILIMIT`, and `ISTRIDE`

☐ Briefly explain each choice.

```
double sum = 0.0;
char *a = 0x2000000;  // sizeof(char) = 1 character.

char *b =                                            // FILL IN

int i;
int ILIMIT =                                         // FILL IN

int ISTRIDE =                                        // FILL IN

for(i=0; i<ILIMIT; i++)
  sum += a[ i * ISTRIDE ]  +  b[ i * ISTRIDE ];
```

Problem 5: Answer each question below.

(*a*) (5 pts) Consider trap instructions and instructions that raise exceptions.

☐ What are trap instructions typically used for?

☐ Sometimes when an instruction in a program raises an exception the program ultimately is allowed to continue. Give an example of such an exception, and what the handler might do.

(*b*) (5 pts) When a MIPS instruction raises an exception the type of exception is written to the cause register. SPARC V8 lacks an equivalent of a cause register, so what does it use as a substitute? Explain.

☐ SPARCs alternative to MIPS' cause register.

(*c*) (5 pts) An early critic might have said that the improvements realized by dynamically scheduled systems could be achieved on much less expensive statically scheduled systems by using better compilers. The particular compiler improvements would help statically scheduled systems but have no impact on dynamically scheduled ones. Consider two-way superscalar systems for the examples needed below.

☐ Explain what the compilers would have to do and why.

☐ Provide an example, showing code before and after optimization.

(*d*) (5 pts) What is it about loads that allow dynamically scheduled systems to outperform statically scheduled systems even with good compilers?

☐ Explain.