Name _____

Computer Architecture
EE 4720
Final Examination

13 May 2004, 17:30–19:30 CDT

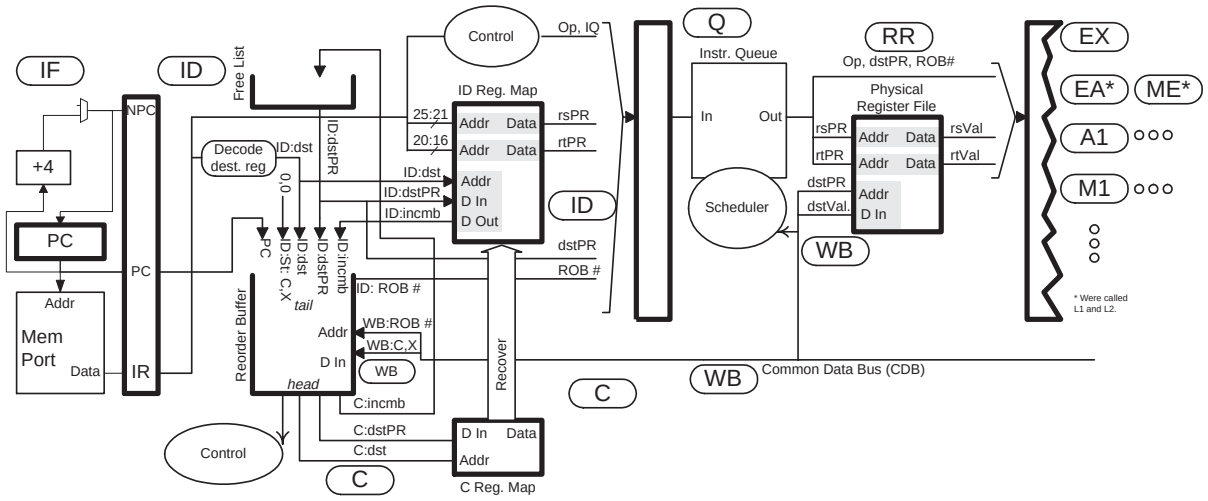
Problem 1	_____	(19 pts)
Problem 2	_____	(18 pts)
Problem 3	_____	(19 pts)
Problem 4	_____	(19 pts)
Problem 5	_____	(25 pts)
Exam Total	_____	(100 pts)

Alias _____

Good Luck!

Problem 1: The MIPS code below executes on the illustrated implementation. (19 pts)

- The implementation makes backup copies (not illustrated) of the ID map for each predicted branch instruction.
- The implementation is scalar, but there can be any number of writebacks per cycle.



The pipeline execution diagram on the next page shows the complete execution of the first instruction, a branch, and partial execution of the others. (Because of instructions before it, the branch enters `RR` after a wait of four cycles and commits after waiting another two.) The branch is mispredicted and some instructions will have to be squashed.

(*a*) Complete the pipeline execution diagram.

☐ Show where instructions are squashed.

☐ Show correct path instructions.

☐ Show each instruction until it is squashed or commits.

(*b*) Complete the tables:

☐ Physical register file. Assume the result of the load is 101, `xor` is 102, and `add` is 103. You can make up your own physical register numbers!

☐ Show where registers are removed from and put back in the free list.

☐ Complete the ID map. Don't forget to include the effect of branch misprediction recovery.

☐ Complete the commit map.

Problem 1, continued: Continued from previous page.

```
 # Cycle              0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16

 bneq r1, r2 SKIP    IF  ID  Q                   RR  B   WB          C

 lw r3, 0(r4)            IF  ID  Q   RR  EA              ME  WB

 xor r3, r8, r6              IF  ID  Q   RR  EX

SKIP:

 add r3, r3, r7                  IF  ID  Q   RR
 # Cycle              0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
 # ID Map
```

```
 # Cycle              0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
 # Commit Map
```

```
 # Cycle              0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
 Physical Register file
```

```
 # Cycle              0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
```

Problem 2: Answer each question about the illustrated MIPS implementation. (18 pts)

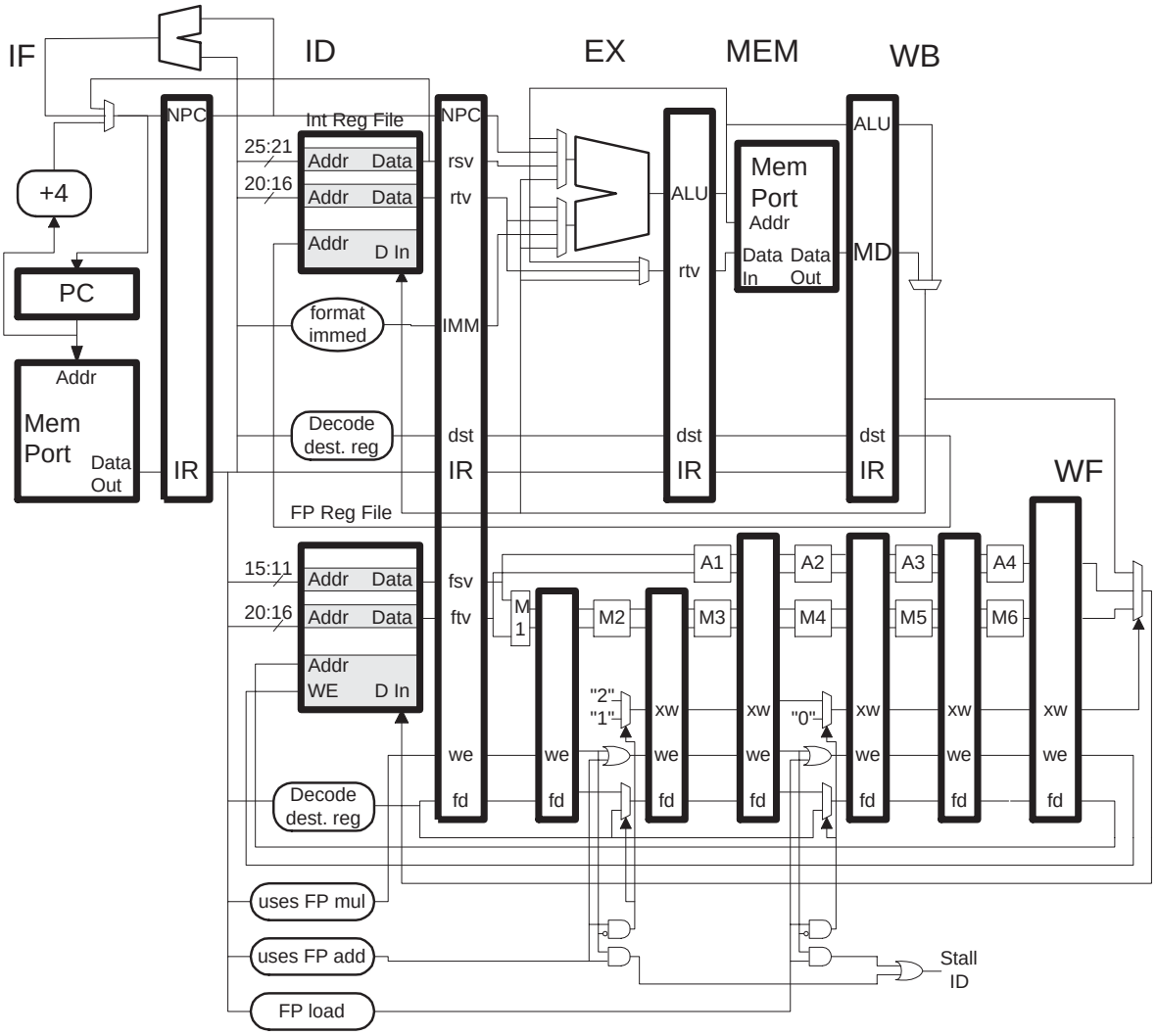
(*a*) Complete a pipeline execution diagram for the code below running on the illustrated implementation. Assume that any needed bypass connection is available and please check for dependencies.

```
# Cycle              0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16

add.d f6, f2, f4

add.d f4, f6, f2
```

(*b*) In the diagram below add the bypass connection(s) needed for the code above. **Do not** add bypass connections that are not needed. (Don't worry if it's a tight squeeze in the diagram, but be legible.)

(*c*) Add control logic that generates the stall signal encountered in the code above.

(*d*) Add control logic for the bypass multiplexor added in a previous part.



Problem 3: The following questions refer to the assembly language code below. Branch outcomes are shown. (19 pts)

```
LOOP:
 0x1000:
B1: beq r1,r2 SKIP1     T  T  T  N  N  T  T  T  N  N  T  T  T  N  N
 nop                                                              !
 add r5, r6, r7                                                   A
SKIP1:                                                            !
 # **Ten** arithmetic instructions only until next branch, no other instructions.
 #                                                                !
B2: beq r3,r4 SKIP2      T  N  T  N  T  N  T  N  T  N  T  N  T  N  T  N
 nop
 add r8, r9, r10
SKIP2:
 add r11, r12, r13
 j LOOP
 nop
```

(*a*) Suppose the code above runs on a system using a bimodal branch predictor with a 256-entry branch history table. What would be the best prediction accuracy of branch `B1` and `B2` after warmup. *Hint: "Best" applies to* `B2` *but not* `B1`.

☐ Accuracy of `B1`.

☐ Accuracy of `B2`.

☐ Why is there a "best" accuracy?

(*b*) Suppose the code above runs on a system using a local predictor with a 256-entry BHT. What is the smallest local history size that would allow the two branches above to be predicted with 100% accuracy?

☐ Smallest size. Explain

(*c*) Suppose the code runs on a system using a gshare branch predictor with a 10-bit GHR. Show the contents of the GHR at time `A` (see the right-hand side of the diagram).

☐ Gshare GHR contents:

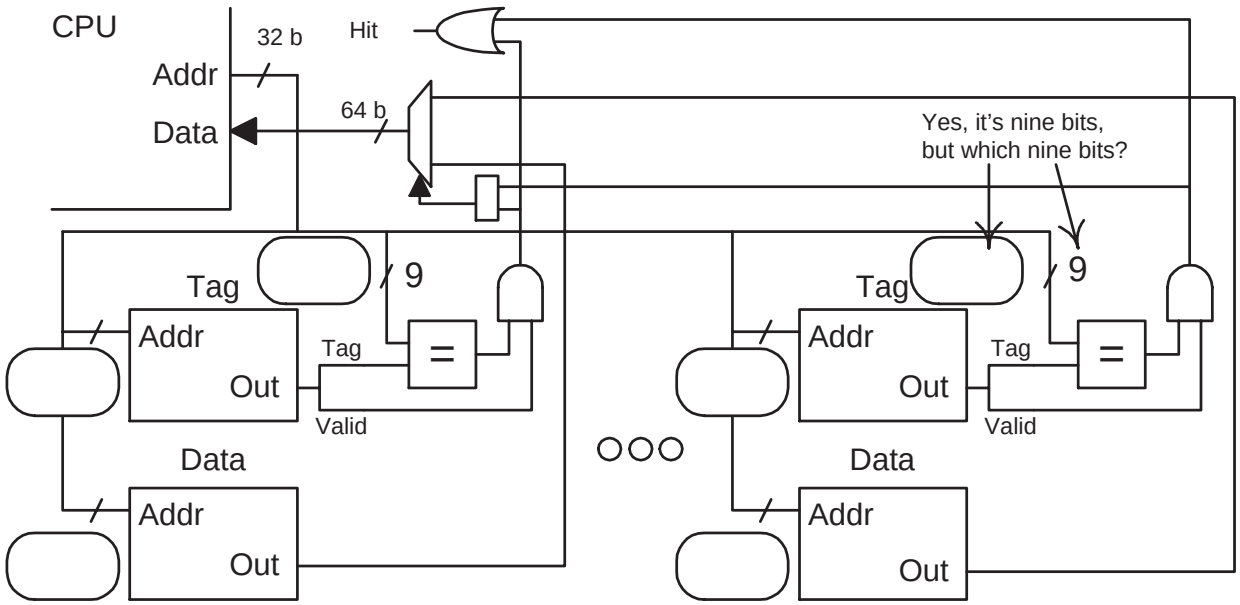
(*d*) Suppose the code runs on a PPC970 (or a POWER4) processor. Show the contents of what it uses for a GHR at time `A`. State any assumptions. *Note: This part based on a homework assigned Spring 2004, and would not be asked other semesters.*

☐ PPC970 GHR contents:

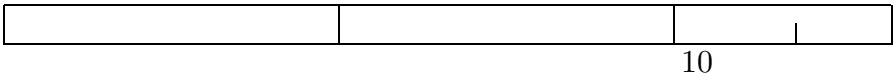
Problem 4: (19 pts) The diagram below is for a four-way set-associative cache on a system with 8-bit characters.

(*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.)

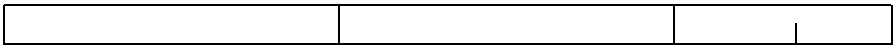
Address: 

☐ Cache Capacity (Indicate Unit!!):

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

☐ Line Size (Indicate Unit!!):

☐ Show the bit categorization for a **direct mapped** cache with the same capacity and line size.

Address: 

Problem 4, continued:

(*b*) The code below runs on the cache from the previous part. When the code below starts running the cache is empty. Consider only accesses to the array and assume the cache is cold (empty) when the code starts.

```
short int *a = 0x1000000;  // sizeof(short int) = 2 characters
int sum, i, j;
int ILIMIT = 1024;

for(j=0; j<2; j++)
  for(i=0; i<ILIMIT; i++)
    sum += a[ i ];
```

☐ What is the hit ratio for the program above?

(*c*) Choose values for the address of `b`, `ILIMIT`, and `ISTRIDE` so that the cache is completely filled with the minimum number of accesses. *Note: The original exam read "minimum number of misses," is easier to do.* The address of `b` must be greater than `a` and as small as possible.

```
short int *a = 0x1000000;  // sizeof(short int) = 2 characters
int sum, i;


short int *b =


int ILIMIT =


int ISTRIDE =

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

Problem 5: Answer each question below.

(*a*) One way to improve the performance of an implementation is to redesign the processor so that it uses more stages. Increasing the number of stages beyond five will yield a big improvement in performance, but at some point adding stages will have little effect.(5 pts)

☐ Give a reason for this limit having to do with the pipeline latches.

☐ Give a reason for this limit having to do with program characteristics.

(*b*) An instruction does not raise precise exceptions (because the ISA says it does not have to). (5 pts)

☐ Name something its handler cannot do (but could do if the exception were precise).

(*c*) The code below uses an indirect load. Re-write it using MIPS instructions. (5 pts)

```
lw r1, @(r2)   # Load using indirect addressing.
```

(*d*) Every integer instruction that reads a GPR uses the `rs` and `rt` fields (or just one of them). What would be the disadvantage of a modified ISA in which some instructions use other fields to specify GPR source registers? (5 pts)

(*e*) Consider two options for the design of a load/store unit (LSU) for a dynamically scheduled system. The aggressive option would execute the code below quickly, while the conservative option would execute it more slowly. *Hint: the conservative option is how the LSU was described in class. Note: The original problem used* `r1` *for the address, which was not intended, but the answer with* `r1` *is similar.*

(5 pts)

```
div.d f2, f4, f6
sw r1, 0(r9)
sh r2, 0(r9)
sb r3, 0(r9)
lw r4, 0(r9)
```

☐ What is it about that code that requires special treatment?

☐ Describe how the aggressive option would execute the code.

☐ Give a brief argument for the conservative option, feel free to include made-up data (for the purposes of this test only!).