Full Name:______________________________________

Andrew ID:______________________________________

CS 15-347, Spring 1995

Final Exam

May 15, 1995

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name and Andrew ID *on every indicated sheet of the exam.*
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maximum score of 71(!) points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like.

Do not write below this line

1. ________	4. ________	7. ________	10. ________
2. ________	5. ________	8. ________	11. ________
3. ________	6. ________	9. ________	12. ________
Total.________			

Full Name:____

Andrew ID:____

Absolute Value, Revisited

Problem 1. (5 points):

Consider the following MIPS code sequence:

```
sra  $1,$10,31     # Shift right arithmetic
xor  $11,$10,$1    # Bit-wise Exclusive Or
subu $11,$11,$1    # Subtract
```

Recall that with an *arithmetic* right shift, the bits are shifted right, but filling in with the sign bit, rather than with 0.

We claim that the effect of this sequence is to set register `$11` to the absolute value of `$10` for *almost all* value(s), assuming a two's complement representation of signed integers.

A. To understand this code better, try executing the sequence with a word size of 4 bits, for the following cases. Of course, the shift should be by 3 bits, not 31.

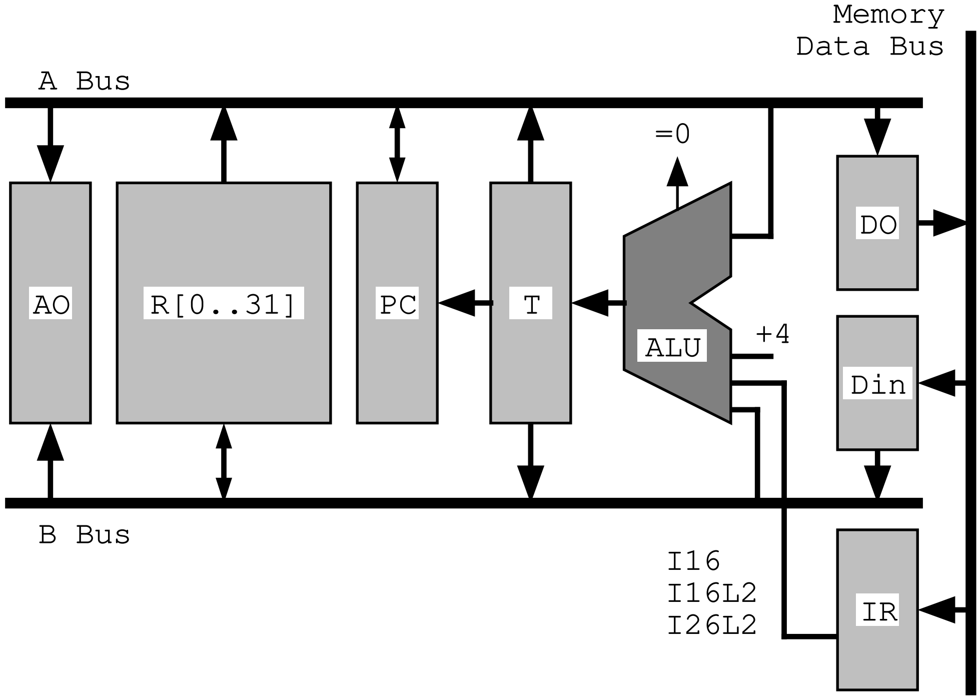
Decimal	Steps in binary				Decimal
Initial	Initial	Shifted	Xor'ed	Subtracted	Result
3					
-1					
7					
-7					

B. Explain briefly (in English) why this sequence has the desired effect.

C. For what value(s) is the final result not equal to the absolute value of the original?

NOP Instructions and Branch Delay Slots

The following problems concern the implementation and use of NOP (no operation) instructions for uMIPS, the microcoded implementation of a MIPS subset. Recall that uMIPS implements delayed branches. For your reference, here is a diagram of the data path.



Here are the state sequences for some instructions relevant to the problem. In determining the execution time for an instruction, it is assumed that each state requires one clock cycle.

FUN

Rs->A->ALU	
Rt->B->ALU	Fun
T->PC	
PC->A->ALU,AO	IR
+4->ALU	Add
T->B->Rd	Disp

ADDI

Rs->A->ALU	
I16->ALU	Add
T->PC	
PC->A->ALU,AO	IR
+4->ALU	Add
T->B->Rt	Disp

Load

Rs->A->ALU	
I16->ALU	Add
T->PC	
	Read
T->B->AO	
PC->A->ALU,AO	IR
+4->ALU	Add
Din->B->Rt	Disp

Store

Rs->A->ALU	
I16->ALU	Add
T->PC	
Rt->A->DO	Write
T->B->AO	
PC->A->ALU,AO	IR
+4->ALU	Add
	Disp

Full Name:_______________________________________

Andrew ID:_______________________________________

Problem 2. (5 points):

The MIPS instruction set does not include an explicit NOP operation.

A. Describe how an existing instruction could be used to have an equivalent effect, requiring two cycles on uMIPS.

B. Fill in the template below to show how you could implement NOP in a single clock cycle on uMIPS. Your implementation must follow the conventions for the uMIPS implementation, namely at the start of each instruction execution:

- The instruction register is loaded with the instruction to be executed.
- The program counter `PC` contains the address of the instruction to be executed.
- The temporary register `T` contains the address of the next instruction to be executed. This will either be `PC` + 4, or the address of a jump or branch target.

NOP

Full Name:_____________________________________

Andrew ID:_____________________________________

Problem 3. (6 points):

Suppose you are a compiler writer, trying to optimize the code generation for programs running on uMIPS. Recall that uMIPS follows the MIPS convention on delayed branches for the sake of code compatibility. You have determined that you could fill 70% of the executed branch delay slots with load or store instructions, while the rest must be filled with NOP's. Of these loads and stores, only 60% of would do "useful" work, while the other 40% would load or store values that are not needed for the particular execution sequence. Assume you must make a choice between one of the following two conventions:

Fill Generate code that fills these delay slots

No Fill Insert NOP instructions in the delay slots, and put the loads and stores elsewhere, such that they will only be executed when needed.

Your task is to present quantitative arguments as to whether or not this optimization should be performed under two conditions: the first when you must use a two cycle instruction to create the effect of a NOP and the second when you have a single cycle NOP instruction available.
Here is the way you can make your argument. Suppose a program executes 100 branches. Determine the number of clock cycles wasted doing either useless load/stores or NOP's. Fill in the tables below to determine which variant wastes the fewest cycles.

A. You must use a two cycle instruction to create the effect of a NOP.

Category	Fill	No Fill
Useless Load/Stores		
NOP's		
Total		

Which option would you choose?

B. You have a single cycle NOP instruction available.

Category	Fill	No Fill
Useless Load/Stores		
NOP's		
Total		

Which option would you choose?

MIPS Registers and Stack Management

The following problem concerns the conventions for saving and restoring registers in MIPS.
An optimizing compiler is given the following C code:

```
mangle(int val) {
  if (val & 0xf)
    return val;
  else
    return val & mangle(tweak(val));
}
```

The compiler generates the following partial assembly code, missing any necessary instructions for stack frame management. It has not yet been determined whether the register $X will be instantiated as a callee- or caller-save register.

```
    mangle:
            move    $X,$4          # $X is destination
            andi    $2,$X,0x000f
            BNE     $2,$0,$L2
            #nop
            JAL     tweak
            #nop
            JAL     mangle
            move    $4,$2
            J       $L4
            and     $2,$X,$2
    $L2:
            move    $2,$X
    $L4:
            J       $31
            #nop
```

Note the following:

- All branches have delay slots, and they have already been filled, i.e., no instructions will be re-ordered by the assembler [As signified by the capitalized operations.]
- The instruction “move” will be instantiated into some appropriate MIPS instruction.

Full Name:_______________________________________

Andrew ID:_______________________________________

Problem 4. (8 points):

Answer the following questions according to *MIPS procedure calling conventions*.

Specify requested locations in the code as *between two consecutive line numbers* in the given code, e.g., write "54 – 55," to signify between lines 54 and 55. For some questions, there are multiple correct answers. Don't worry about the relative ordering of instructions that are inserted between the same two lines in the given code.

A. Where, if necessary, should the stack frame be allocated?

   Popped?

B. Are there any registers, excluding $X, which will need to be saved on the stack at some point during the execution of this routine?

   If so, which one(s), and where should each be saved and restored?

C. If $X were instantiated as a callee-save register, would it need to be saved on the stack?

   If so, where should it be saved? Restored?

D. If $X were instantiated as a caller-save register, would it need to be saved on the stack?

   If so, where should it be saved? Restored?

Pipelined Processor Implementation

You are a part of the processor design team at the TLA corporation. This design team is developing a pipelined implementation of a *variant* of the MIPS instruction set. The instruction set contains the following instructions: R-type (register-register), `LW`, `SW` and `BEQ`. The processor does not have delay slots in load and branch instructions.
The pipeline is a *four stage pipeline* with the stages

IF Instruction Fetch stage.

ID Instruction Decode, Register Fetch stage.

EXMEM Execute and Memory operation stage.

WB Register File writeback stage.

Each pipeline stage takes one clock cycle. For the most part, this pipeline is very similar to the MIPS pipeline. Some *important differences* between the MIPS pipeline and this pipeline are listed below:

- The **EXMEM** stage does memory access for data in case of a LW or SW instruction and, unlike the original MIPS pipeline, there is no separate **MEM** pipeline stage.
- With the help of extra hardware, the branches have been optimized to take fewer cycles.
- In the **ID** stage after the register values are obtained, the memory address is calculated and stored in MAdr. In this pipeline stage the branch target address (BTA) is also computed.

The pipeline has the following *hardware resources*:

ALU This is only used to perform the operations specified in R-type instructions.

Adders $+_{pc}$, $+_{imm}$ and $+_{bta}$. The pipeline table on the next page shows where these adders are used.

1. $+_{pc}$ increments the PC value by 4 in the IF stage.
2. $+_{imm}$ is used to compute the memory address in the ID stage.
3. $+_{bta}$ is used to compute the branch target address (BTA) in the ID stage (the pipeline description given to you shows this).

Register File This has three ports, two read and one write. In a single cycle two read operations and a write operation can be done and the write operation precedes the read operation.

Cache Unified instruction and data cache from which the processor fetches instructions and does data access. A memory read or write operation takes one cycle. The memory system can support only *one* operation per cycle.

The table below describes the pipeline. Some of the notation used in the table is given below.

IR $\leftarrow$ Memory[PC] Set the instruction register (IR) to the word at the memory address in the program counter (PC).

PC $\leftarrow$ PC $+_{pc}$ 4 Increment the program counter by 4. Use the $+_{pc}$ adder for this operation.

Rs Register[IR[25:21]]

Rt Register[IR[20:16]]

Rd Register[IR[15:11]]

Imm <IR[15]ˆ 17 # IR[14:0]> = (sign extended immediate field)

ImmS2 <IR[15]ˆ 15 # IR[14:0] # 00> = (sign extended immediate field shifted left by 2)

STAGE	R-Type Instruction	LOAD Instruction	STORE Instruction	BEQ Instruction
IF	IR $\leftarrow$ Memory[PC] PC $\leftarrow$ PC $+_{pc}$ 4	IR $\leftarrow$ Memory[PC] PC $\leftarrow$ PC $+_{pc}$ 4	IR $\leftarrow$ Memory[PC] PC $\leftarrow$ PC $+_{pc}$ 4	IR $\leftarrow$ Memory[PC] PC $\leftarrow$ PC $+_{pc}$ 4
ID	A $\leftarrow$ Rs B $\leftarrow$ Rt	A $\leftarrow$ Rs B $\leftarrow$ Rt MAdr $\leftarrow$ A $+_{imm}$ Imm	A $\leftarrow$ Rs B $\leftarrow$ Rt MAdr $\leftarrow$ A $+_{imm}$ Imm	A $\leftarrow$ Rs B $\leftarrow$ Rt BTA $\leftarrow$ PC $+_{bta}$ ImmS2 if (A==B) PC $\leftarrow$ BTA
EXMEM	ALUout $\leftarrow$ A op B	MemData $\leftarrow$ Memory[MAdr]	Memory[MAdr] $\leftarrow$ B	
WB	Rd $\leftarrow$ ALUout	Rt $\leftarrow$ MemData		

The pipeline above has *hazards* which prevent the next instruction in the instruction stream from executing during its designated clock cycle. Hazards reduce the performance from the ideal speedup gained by pipelining. There are three classes of hazards:

Structural These arise from resource conflicts when the hardware cannot support all possible combinations of instructions in simultaneous overlapped execution.

Data These arise when an instruction depends on the results of a previous instruction in a way that is exposed by the overlapping of instructions in the pipeline.

Control These arise from the pipelining of branches and other instructions that change the PC.

A number of hardware techniques can be used to minimize the performance loss due to pipeline hazards.

Full Name:_______________________________________
Andrew ID:_______________________________________

Problem 5. (5 points):

Your responsibility as a part of the design team is to identify structural, data and control hazards in the pipeline and suggest solutions to them. The solution you suggest should minimize the performance loss due to these hazards.

A. There is one structural hazard in the design which prevents an instruction fetch on every cycle. Identify it and propose a solution to eliminate it.

B. Upon encountering a branch the processor continues fetching instructions assuming that the branch is not taken. If the branch is indeed taken, it flushes the instructions in the pipe which have been fetched assuming a not taken branch.

   What is the branch penalty for a branch when it is taken ? When it is not taken ?

   If an application contains 10% branches and 60% of the branches are taken, and all non-branch instructions take one cycle, what is the processor CPI for this application?

C. For this problem consider the 5-stage pipeline as described in Fig 6.48 of your book and the 4-stage pipeline described above. Compute the CPI of the two processors for an application with 20% loads. In this application there is a probability of 0.1 that the value read by the load instruction is used by the instruction following the load. Assume all other instructions require one cycle.

   5-stage CPI:

   4-stage CPI:

Full Name:______________________________________

Andrew ID:______________________________________

Problem 6. (14 points):

There are a number of data hazards in the design which will cause the pipeline to operate incorrectly for some instruction sequences. You must identify these data hazards and a solution to fix the hazard by filling in the table entries below: For full credit, each row of a column must be correctly and completely filled. In case a hazard does not exist, you may strike out columns 3 to 9. For fixing the hazard, you must indicate a solution which minimizes the speed penalty.

In column 7 indicate the latest pipe stage at the beginning of which the operand must be available.

MIPS CODE	Hazard Exists? (Y/N)	Hazard Type: Value — Register Number	Where Generated? — Instr Num.	Where Generated? — Pipe Stage	Where Needed? — Instr Num	Where Needed? — Pipe Stage	Hazard Fix: Stall (Y/N)	Hazard Fix: Forward (Y/N)
1. add $1,$2,$3 2. add $3,$1,$2								
1. add $1,$2,$3 2. lw $5,0xf0($1)								
1. add $1,$2,$3 2. sw $1,0xf0($5)								
1. add $1,$2,$3 2. add $3,$5,$6 3. add $5,$1,$2								
1. lw $1,0x40($2) 2. add $3,$1,$6								
1. lw $1,0x40($2) 2. sw $1,0x44($3)								
1. lw $1,0x40($2) 2. add $0,$0,$0 3. sw $1,0x44($3)								

Cache Line Size

The following problem concerns the relation between structure access and caches.
Suppose we are trying to decide on the cache line size for our latest uniprocessor machine. We hope to pick the number of words L in a line to optimize the program `sum_list`. Below is the C code from Homework H1.

```
/* Declare a linked list data type */
typedef struct list_t *list_ptr;

typedef struct list_t {
  int val;
  list_ptr next;
}  list_ele;

/* Compute the sum of the values in a linked list */
int sum_list(list_ptr ls)
{
  int val = 0;
  while (ls != NULL) {
    val += ls->val;       (*)
    ls = ls->next;        (**)
  }
  return val;
}
```

Full Name:____

Andrew ID:____

Problem 7. (6 points):

We will assume that the linked list fits within main memory, but it is too large and "mixed" so that every read in line (*) causes a cache miss. Thus, the first read, line (*), will have a cache miss rate of 1.0 by assumption. The read in line (**) may or may not cause a cache miss, and hence its miss rate will be less than 1.0.

We will assume the time to fetch a cache line is $d + r \cdot L$ where d is the "delay overhead" and r is the "word transfer time" for transmitting words from memory to cache. We also assume that when there is a cache miss, the processor must stall until the entire line is filled.

A. Diagram the 4 possible alignments of a structure `list_t` relative to the cache line boundaries for $L = 4$?

B. What is the miss rate for the second read, line (**), as function of L?

C. Give an expression for the expected time to perform the two reads from lines (*) and (**), in terms of r, d, and L.

D. Compute the value of L which minimizes the expected time to perform the two reads in (*) and (**). Your answer should be an expression in d and r.

E. If the delay overhead is $d = 90$ cycles and the word transfer time is $r = 10$ cycles per word, what line size would minimize the expected time to read?

F. Would this make a good choice for line size? Explain.

Full Name:_________________________________

Andrew ID:_________________________________

Problem 8. (6 points):

Recall from lecture that we studied a cache system for a shared memory multiprocessor where each line on each cache can be in one of four states

State ID	State Name	Relation to memory
IN	Invalid	
EC	Exclusive-Clean	Unmodified
SC	Shared-Clean	Unmodified
ED	Exclusive-Dirty	Modified

[This protocol is similar to the one described in Chapter 9 of the book (Fig. 9.7), but in that protocol, states EC and ED are merged.]

The two requests by the processor that affect the state of a cache line are reads and write on a single word. We say that a processor read (respectively, write) request is a read (resp., write) *cache miss* when it causes the cache to generate bus traffic.

The three commands a cache can issue on the memory bus are:

- write line to memory
- read line from memory or other cache
- invalidate line in other caches

These requests may affect the state of other cache lines.

A. What state or states can a cache line be in when a cache read miss occurs?

B. When a cache write miss occurs?

C. If some cache has a line in state ED what state or states may some other cache have for this line?

D. Describe under what conditions the invalidate bus command would be used.

E. Suppose the processor requests a write to a line that is in state IN. Describe the sequence of states and bus commands the processor's cache would go through in order to satisfy this request.

Memory Hierarchy

The following series of problems concern a hierarchical memory structure with the following features:

- The memory is byte addressable.
- Virtual addresses are 16 bits wide.
- Physical addresses are 14 bits wide.
- The page size is 128 bytes.
- The TLB is 4-way set associative with 16 total entries.
- The cache is direct mapped, with a 4 byte line size and 16 total lines.

In the following tables, all numbers are given in hexadecimal. The contents of the TLB are as follows:

TLB

Index	Set 0			Set 1			Set 2			Set 3		
	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	02	13	1	04	32	1	00	28	1	01	0C	0
1	03	2A	1	01	37	0	02	0A	0	02	0C	1
2	01	38	0	00	18	1	08	01	0	03	11	1
3	05	2B	1	04	1A	1	0D	07	0	01	10	1

The page table for the first 16 pages, and the cache contents are as follows

Page Table (Partial)

VPN	PPN	Valid	VPN	PPN	Valid
000	28	1	008	13	1
001	42	0	009	0C	1
002	18	1	00A	07	1
003	0B	1	00B	24	0
004	0C	0	00C	05	0
005	0B	0	00D	2A	1
006	43	1	00E	11	1
007	10	1	00F	11	0

Cache

Index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	19	1	99	11	23	11
1	15	0	55	59	0B	41
2	13	0	12	E2	89	A4
3	22	1	3A	DD	83	11
4	20	1	45	48	0C	A1
5	22	1	23	99	8E	BB
6	2E	0	29	90	E2	59
7	16	1	62	0C	FF	22
8	22	1	5D	13	E3	49
9	13	1	0D	2B	02	3F
A	1F	1	89	38	F4	CC
B	C4	0	40	A7	05	6F
C	B1	1	38	5F	B1	A4
D	26	1	A7	31	37	F8
E	99	0	0D	0C	52	19
F	51	1	88	96	0B	D8

Full Name:____
Andrew ID:____

Problem 9. (4 points):

A. The box below shows the format of a virtual address. Indicate (by labeling the diagram) the fields that would be used to determine the following:

VPN The virtual page number

VPO The virtual page offset

TLBI The TLB index

TLBT The TLB tag

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

B. The box below shows the format of a physical address. Indicate (by labeling the diagram) the fields that would be used to determine the following:

PPN The physical page number

PPO The physical page offset

CO The byte offset within the cache line

CI The cache index

CT The cache tag

13	12	11	10	9	8	7	6	5	4	3	2	1	0

Full Name:_____________________________________

Andrew ID:_____________________________________

For the following problems, you are to show how the memory translates a virtual address into a physical and accesses the cache. For the given virtual address, indicate the TLB entry accessed, the physical address, and the cache byte value returned. Indicate whether the TLB misses, whether a page fault occurs, and whether a cache miss occurs. Carry each translation as far as possible, i.e., until there is insufficient data to complete the translation. Mark '—' through entries that cannot be correctly determined.

Problem 10. (5 points):

Virtual address `04C1`

A. Virtual address format

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

B. Address translation

Parameter	Value
VPN	
TLB Index	
TLB Tag	
TLB Hit? (Y/N)	
Page Fault? (Y/N)	
PPN	

C. Physical address format

13	12	11	10	9	8	7	6	5	4	3	2	1	0

D. Physical memory reference

Parameter	Value
Byte offset	
Cache Index	
Cache Tag	
Cache Hit? (Y/N)	
Cache Byte returned	

Full Name:______________________________________

Andrew ID:______________________________________

Problem 11. (5 points):

Virtual address `02F0`

A. Virtual address format

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

B. Address translation

Parameter	Value
VPN	
TLB Index	
TLB Tag	
TLB Hit? (Y/N)	
Page Fault? (Y/N)	
PPN	

C. Physical address format

13	12	11	10	9	8	7	6	5	4	3	2	1	0

D. Physical memory reference

Parameter	Value
Byte offset	
Cache Index	
Cache Tag	
Cache Hit? (Y/N)	
Cache Byte returned	

Full Name:_____________________________________

Andrew ID:_____________________________________

Problem 12. (5 points):

Virtual address `0569`

A. Virtual address format

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

B. Address translation

Parameter	Value
VPN	
TLB Index	
TLB Tag	
TLB Hit? (Y/N)	
Page Fault? (Y/N)	
PPN	

C. Physical address format

13	12	11	10	9	8	7	6	5	4	3	2	1	0

D. Physical memory reference

Parameter	Value
Byte offset	
Cache Index	
Cache Tag	
Cache Hit? (Y/N)	
Cache Byte returned	