Your name: ____________________

EXAM #1

CS 2410 Graduate Computer Architecture

Spring 2016, MW 11:00 AM – 12:15 PM

Directions: This exam is closed book. Put all materials under your desk, including cell phones, smart phones, smart watches, headphones, laptops, tablets, e-readers, etc. All questions are marked with their point value. There should be plenty of workspace provided in the exam booklet, but if you need extra pages, you may use blank pieces of paper.

Show work: Be sure to show all work and turn in any extra pages that you use. If you do not show your work, you may not receive full or partial credit for a correct or wrong answer. Write legibly. If your handwriting cannot be read, then you will not receive credit for an answer.

5-stage MIPS pipeline: Recall that the MIPS 5-stage pipeline is Fetch (IF), Decode/Register Read (ID), Execute (EX), Memory (MEM) and Writeback (WB). You should assume for all questions, unless stated otherwise, the pipeline has full forwarding, branches are resolved in the Decode/Register Read stage (i.e., the branch comparison is done here). Be careful: When needed, the problems will state whether to assume a delayed branch or branch prediction.

Your name: ____________________

1. *20 points* each. write "T" or "F" to indicate whether the statement is true or false.

   ___ A RISC instruction set usually has variable length instructions.

   ___ A RISC instruction set usually has displacement and scaled addressing modes.

   ___ An accumulator instruction set usually minimizes memory accesses.

   ___ A register-memory instruction set architecture is typical of CISC.

   ___ A typical CISC instruction set permits only two source operands.

   ___ Branches usually use PC-relative addressing rather than absolute addressing.

   ___ Instruction-level parallelism is a form of task-level parallelism.

   ___ SIMD execution is a form of data-level parallelism.

   ___ The number of anti- and output dependences limits the loop unroll factor.

   ___ Unrolling a loop cannot harm program performance.

2. The CPU time equation is CPU time = IC + CPI + CC. For each term in the equation, explain what aspects of the computer architecture or the microarchitecture affect the term.

   - *3 points*. IC (instruction count):

   - *4 points*. CPI (clock cycles per instruction):

   - *4 points*. CC (clock cycle time):

3. *7 points*. Let's consider a 4-stage pipeline that has the stages Fetch, Decode/Read, Execute, Writeback. Branches are resolved (i.e., the branch comparison is computed) in Decode/Read. Loads and stores compute their effective address in Execute and access memory in Writeback. Assume the instruction set is the same as MIPS. Where is forwarding needed in this pipeline?

   You may answer by giving a list of forwarding paths: indicate the producer and consumer stages where you need to forward from/to. Alternatively, you may draw the pipeline with all forwarding paths clearly shown.

4. *6 points*. Let's again consider the same pipeline from the previous problem. Where are pipeline interlocks required? Recall that an interlock is a situation where a hazard must be handled by stalling the pipeline. In answering this question, list the pipeline stage involved in the interlock and briefly explain why an interlock is required.

5. Consider the classic 5-stage MIPS pipeline (Fetch, Decode, Execute, Memory, Writeback). Suppose branch prediction is used (there is no delay slot instruction); the branch target address is known in the Fetch stage and a branch can be resolved in Decode, where the registers are read. The branch prediction (correct) accuracy is 80% for the code below. Further assume the clock cycle time is 1000 MHz; there is full forwarding; indexed addressing mode is available (reg + reg = effective address); and, an instruction needs only 1 cycle in Execute (including the `mul`). The number on the left of the code is an instruction number.

```
         ;; for (i = 0; i < 10; i++)
         ;;    A[i] = A[i] + A[i] * k;
         ;; assume registers are initialized prior to loop
         ;; R1 is k, R2 is base address of A
         ;; R4 is index i (as word, i.e., i * 4)
00 loop: lw      R5,(R2+R4)     ; R5 = A[i]
01       mul     R6,R5,R1       ; R6 = R5 * k
02       add     R6,R5,R6       ; R6 = R5 + R6
03       sw      R6,(R2+R4)     ; A[i] = R6
04       addi    R4,R4,4        ; increment word offset for index i
05       sub     R7,R4,40       ; iterate 10 times (10 * 4 bytes = 40)
06       bnez    R7,loop        ; branch to loop if R5 != 0
```

For the loop above, answer:

- *2 points*. What is the instruction count?

- *4 points*. What is the CPI?

- *4 points*. What is the CPU time?

6. *10 points*. Suppose the 5-stage MIPS pipeline is changed into a 7-stage pipeline. In the 7-stage pipeline, the original Decode stage is split into 2 stages: Decode, followed by Register Read. The original Memory stage is split into 2 separate stages: Access, followed by Data Delivery. A read data value is available after Data Delivery.

   Assume there are forwarding paths between all producer and consumer stages with two exceptions. First, a store (`sw`) can execute in EX without the source value to be stored. The effective address can be computed in EX but the register value to be stored can not be forwarded to EX from a later stage. Second, there is no forwarding path from Access back to Access. When forwarding cannot be done, the pipeline stalls until the hazard clears.

   Fill in the table with the movement of the instructions. In the table, use the abbreviations in parentheses: Fetch (F), Decode (D), Register Read (R), Execute (E), Access (A), Data Delivery (DD), Writeback (W), Stall (S). The first instruction is done as an example. Complete only the 15 cycles shown.

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
`lw R5,(R2+R4)`	F	D	R	E	A	DD	W								
`mul R6,R5,R1`															
`add R6,R5,R6`															
`sw R6,(R2+R4)`															
`addi R4,R4,4`															
`sub R7,R4,40`															
`bnez R7,loop`															

7. *5 points*. For your completed table, explain why the pipeline is stopped for any of the stalls that you might have listed.

8. *6 points*. Consider the code below and indicate the instructions (01 to 07) that are parallel with respect to one another. Two parallel instructions i and j can be executed as i before j or j before i. In the table, write "Y" if a pair of instructions are parallel. Otherwise, write a "N". The number on the left of the code is the instruction number used in the table.

```
        ;; int A[12];
        ;; for (i = 1 to 10) do {
        ;;    A[i] = A[i] + A[i+1];
        ;;    A[i-1] = 2*A[i+1] + k; }
        ;; assume R1 is k, R2 is address A[i], R3 is address A[10]
01 loop: lw     R5,0(R2)       ; R5 = A[i]
02       lw     R6,4(R2)       ; R6 = A[i+1]
03       add    R5,R5,R6       ; R5 = R5 + R6
04       sw     R5,0(R2)       ; A[i] = R5
05       add    R6,R6,R6       ; R6 = 2*A[i+1]
06       add    R6,R6,R1       ; R6 = R6 + k
07       sw     R6,-4(R2)      ; A[i-1] = R6
08       sub    R5,R3,R2       ; check for last array element
09       addi   R2,R2,4        ; increment A address by word size
10       bnez   R5,loop        ; branch to loop if R5 != 0
11       nop
```

	01	**02**	**03**	**04**	**05**	**06**	**07**
01	N						
02		N					
03			N				
04				N			
05					N		
06						N	
07							N

9. *6 points*. Consider the code from the previous problem. Using register renaming, rewrite instructions 01 to 10 to remove all name dependences. Assume registers R15 to R31 are available to use for renaming. When you rename registers, start with R15. Do not reorder the instructions.

10. *10 points*. For your renamed code from the previous problem, schedule the whole loop (instructions 01 to 11) to remove as many stalls as possible. You may reorder/modify the instructions. Assume the 5-stage MIPS pipeline with delayed branches.

11. *9 points*. Consider a 1-bit and a 2-bit predictor. In the table below, indicate the prediction for the branch and whether the preidction is correct . For the 1-bit predictor use the notation "T" for taken, and "NT" for not taken states. For the 2-bit predictor, use "ST" for strongly-taken state, "T" for weakly-taken state, "SNT" for strongly-not-taken state, and "NT" for weakly-not-taken state. Assuming the first entry in the table, fill in the predictor's state **after** branch resolution for each of the remaining branch outcomes.

	1-bit predictor			**2-bit predictor**		
Outcome	**Prediction**	**Next State**	**Correct?**	**Prediction**	**Next State**	**Correct?**
Taken	N	T	No	NT	ST	No
Taken						
Not taken						
Not taken						
Not taken						
Taken						
Not taken						
Taken						
Taken						
Not taken						