Q1. [10 marks] True or false

Determine whether the following statements are true or false. **Circle** the answer. NO further explanation for the answer is needed.

i. A 100MIPS computer is NOT necessarily faster than an 95MIPS computer when they are running the same Java program (MIPS: Million Instructions Per Second)

[✓ True] False

ii. Instruction count in a program depends on the instruction set architecture used.

[✓ True] False

iii. A processor running the Tomasulo's Algorithm may not able to eliminate all the WAR and WAW hazards.

[✓ True] False

iv. The Sun SPARC architecture has fewer addressing modes than IA-32 (x86).

[✓ True] False

v. If we use an instruction in the fall through (untaken) path to fill a delay slot, we must duplicate the instruction.

True [✓ False]

Q2. [18 marks] Multiple choices

Choose ONE answer and write the corresponding letter in the blank space at the beginning of the question.

i. _A__ Which one of the following types of hazards CANNOT be removed by register renaming?

A. RAW hazards
B. WAR hazards
C. WAW hazards
D. None of the above
E. All of the above

ii. _D__ Suppose a superscalar processor has the ability to dispatch one integer and one floating point instructions simultaneously to the integer and floating point functional units and execute them. Which of the following scenarios are IMPOSSIBLE?

A. The processor dispatches 0 integer instruction in one cycle
B. The processor dispatches 1 floating point instruction in one cycle
C. The processor dispatches 2 integer instructions in 2 cycles
D. The processor dispatches 3 floating point instructions in 2 cycles
E. The process dispatches 0 floating point instructions in 3 cycles

iii. _D__ A 1024-entry (4, 2) correlating predictor would use

A. 5001 bits
B. 8192 bits
C. Between 16000 to 17000 bits
D. Between 30000 to 40000 bits
E. More than 100K bits

iv. _B__ Which one of the following is a hazard the basic 5-stage MIPS integer pipeline needs to deal with?

A. WAW
B. RAW
C. RAR
D. WAR

v. _B__ The Itanium computer implements the IA-64 instruction set and belongs to

A. Accumulator based ISA
B. General Purpose Register (GPR) based ISA
C. Stack based ISA

vi. _B__ Which one of the following items does NOT DIRECTLY affect a processor’s ability to exploit ILP? Assume that the computer system has only one single-core processor, only one single-threaded program runs on the system, and the physical memory is large enough to hold all the active pages thus no virtual memory swapping is needed.

A. The design of the branch predictor
B. The clock cycle time
C. The number of instructions a processor can issue every cycle
D. The compiler

Q3. [12 marks] Instruction encoding

A given processor has 32 registers, uses 16-bit immediates, and has 142 instructions (corresponding to 142 operation codes) in its ISA. In a given program, 20 percent of the instructions take one source register and have one destination register; 30 percent of the instructions have two source registers and have one destination register; 25 percent of the instructions have one source register and have one destination register, and take an immediate source operand as well; and the remaining 25 percent have one immediate source operand and one destination register.

a) [6 marks] For each of the four types of instructions, how many bits are required? Assume that the ISA requires that all instructions be a multiple of 8 bits in length, and the operation codes (opcodes) are of fixed length (i.e., the ISA does not use shorter opcodes for some instructions and longer opcodes for others).

8 bits are required to encode 142 instructions ($128 < 142 < 256$).
5 bits are required to encode 32 registers.
And we know that 16 bits are required to encode each immediate.
One source register, one destination register: $8 + 5 + 5 = 18$ bits, which rounds up to 24 bits.
Two source registers, one destination register: $8 + 5 + 5 + 5 = 23$ bits, which rounds up to 24 bits.
One source register, one destination register, and an immediate: $8 + 5 + 5 + 16 = 34$ bits, which rounds up to 40 bits.
One immediate, one destination register: $8 + 16 + 5 = 29$ bits, which rounds up to 32 bits.

b) [6 marks] How much less memory does the program take up if a variable-length instruction set encoding is used as opposed to a fixed-length encoding?

Since the longest instruction type requires 40 bits to encode, the fixed-length encoding will have 40 bits per instruction.
For the variable-length encoding, the average number of bits per instruction is: $24 \times 20\% + 24 \times 30\% + 40 \times 25\% + 32 \times 25\% = 30$ bits.
$(40 - 30) / 40 = 25\%$.
Therefore, the variable-length encoding requires 25 percent less space than the fixed-length encoding for this program.

Instruction examples: In some questions, you may need to read or write MIPS instructions in assembly language. Below are some examples of instructions in assembly:

Load a word from memory address REGS[R2]+100 to R1:	`LW R1, 100(R2)`
Load 64 bits from memory to F0 (double precision):	`LD F0,100(R2)`
Store the word in R3 to memory:	`SW R3, 100(R4)`
Add R6 and R7 and assign the result to R5:	`ADD R5, R6, R7`
Add R6 and an immediate, and assign the result to R5:	`ADD R5, R6, 200`
Add F1 and F2 and assign the result to F3 (single-precision):	`ADD.S F3, F1, F2`
Add F1 and F2 and assign the result to F3 (double-precision):	`ADD.D F3, F1, F2`
Branch to address indicated by Label9 if R8 is not zero:	`BNEZ R8, Label9`

You can use ';' to start comments in a line, e.g.,

```
    LW R1, 100(R2); Here is the comment within one line
```

Q4. [18 marks] Dependences

The logic of a program naturally introduces dependences among instructions, including data dependence (true dependence), name dependence (anti-dependence, output dependence), and control dependence (when the decision of whether to execute an instruction is determined by the outcome of a branch). In some pipelines a specific dependence may result in a hazard that prevents the dependent instruction from moving forward in the pipeline without stalls. In some other pipelines, the same dependence may not result in a hazard. Consider the following MIPS assembly code.

```
1 LD      R1, 45(R2)
2 DADD    R7, R1, R5
3 DSUB    R8, R1, R6
4 OR      R1, R5, R1
5 BNEZ    R7, branch_target
6 DADD    R10, R8, R5
7 XOR         R2, R3, R4
```

Instruction 4 (OR R1, R5, R1) has a true dependence and an output dependence on instruction 1 (LD, R1, 45(R2)). However, in the basic 5-stage MIPS pipeline, these dependences do not result in a hazard because instruction 4 is far enough from instruction 1, and the basic 5-stage MIPS pipeline writes the result back to register file in the last (5th) stage. Similarly, instruction 7 (XOR R2, R3, R4) has a control dependence on instruction 5 (BNEZ R7, branch_target), but this dependence would not introduce a hazard in a 5-stage pipeline if the outcome and target address of a branch are computed in the ID (2nd) stage.

Identify all dependences by type in the above code segment; list the two instructions involved; identify which instruction is dependent; and, if there is one, name the storage location involved.

	Dependence type	Independent instruction	Dependent instruction	Storage location
1 (example)	Data	1(LD)	2(DADD)	R1
2	Data	1(LD)	3(DSUB)	R1
3	Data	1(LD)	4(OR)	R1
4	Name	1(LD)	4(OR)	R1
5	Name	1(LD)	7(XOR)	R2
6	Name	2(DADD)	4(OR)	R1
7	Data	2(DADD)	5(BNEZ)	R7
8	Name	3(DSUB)	4(OR)	R1
9	Data	3(DSUB)	6(DADD)	R8
10	Control	5(BNEZ)	6(DADD)	
11	Control	5(BNEZ)	7(XOR)	

Q5 [8 marks] Pipeline stages

Fill in the pipeline timing chart for the code segment. Assume the pipeline is a 5-stage MIPS pipeline with forwarding from the EX/MEM latch to the input of EX-stage and MEM-stage hardware, and forwarding from the MEM/WB latch to the input of EX-stage and MEM-stage hardware. Use F, D, E, M, W and S for fetch (IF), decode (ID), execute (EX), memory (MEM), write back (WB) and stalls, respectively.

Instruction	Clock Cycles										
	1	2	3	4	5	6	7	8	9	10	11
LOAD R2, (R1)	F	D	E	M	W						
ADD R1, R2, R3		F	D	S	E	M	W				
ADD R4, R1, R5			F	S	D	E	M	W			

Q6. [10 marks] General performance

Before and after an architecture "improvement", a computer's CPU 2006 benchmark scores on three programs, bzip2, gcc, and astar, are as follows. The scores are the ratio of performance to the reference machine Sun SPARC Enterprise M8000.

Benchmark	**Score before improvement**	**Score after improvement**
401.bzip2	39	50
403.gcc	36	40
473.astar	19	16

i. (6 marks) Using geometric mean of the scores on the above three programs to summarize the performance, compute the speedup after the improvement. Write down the equations and intermediate steps when you develop the result. (Hint: SPEC benchmark scores are already ratios against a reference machine, and they specify the performance, not execution time, of a computer. A higher score means a faster computer.)

$$\text{GM before improvement} = \sqrt[3]{39 \times 36 \times 19} = 29.88$$

$$\text{GM after improvement} = \sqrt[3]{50 \times 40 \times 16} = 31.75$$

$$\text{Speedup} = \frac{31.75}{29.88} = 1.06$$

ii. (4 marks) Based on the speedup we get in the last question, is the computer faster after the improvement?

Yes.

Q7. (24 marks) The Tomasulo's Algorithm

Suppose we are applying the Tomasulo's algorithm to implement the out-of-order execution. Suppose the processor has the following **non-pipelined** execution units:

- An FP add unit, with each add/subtract instruction taking 2 cycles in the execution step on the FP add unit
- An FP multiply unit, with each multiply instruction taking 3 cycles in the execution step on the FP multiply unit
- An FP divide unit, with each divide instruction taking 6 cycles in the execution step on the FP divide unit

Assume the pipeline is a single-issue pipeline with no speculation. There are two reservation stations for the multiply unit, but only one reservation station for the add unit and one for the divide unit. Hence, reservation stations are named add1, mult1, mult2, div1, respectively. The following tables show the status of internal structures of the Tomasulo Algorithm in cycle 1 and 3 for the following code:

```
        MULT.S    F8, F0, F2
        ADD.S     F2, F8, F1
        DIV.S     F3, F2, F1
        SUB.S     F8, F2, F0
```

A negative number in the "Exec. Complete" is used to indicate the remaining cycles for this instruction in the execute stage (e.g., -1 means the next cycle is the execution completion cycle for this instruction). **Show the instruction status, the reservation station status and register results status for cycle 4, 7, and 9.**

Cycle 1

Instr. status	Instructions (inst.)	Issue	Exec. Com-plete	Write result	RS status	RS Name	Busy	Op	Vj	Vk	Qj	Qk	A
	`MULT.S F8,F0,F2`	1				Add1	N						
	`ADD.S F2, F8, F1`					Mult1	Y	Mult.s	R[F0]	R[F2]			
	`DIV.S F3, F2, F1`					Mult2	N						
	`SUB.S F8, F2, F0`					Div1	N						
Reg. Status	Name	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
	Qi									Mult1			

Note: R[F0] represents the value in register F0.

Cycle 3

Instr. status	Instructions (inst.)	Issue	Exec. Com-plete	Write result	RS status	RS Name	Busy	Op	Vj	Vk	Qj	Qk	A
	`MULT.S F8,F0,F2`	1	-1			Add1	Y	Add.s		R[F1]	Mult1		
	`ADD.S F2, F8, F1`	2				Mult1	Y	Mult.s	R[F0]	R[F2]			
	`DIV.S F3, F2, F1`	3				Mult2	N						
	`SUB.S F8, F2, F0`					Div1	Y	Div.s		R[F1]	Add1		
Reg. Status	Name	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
	Qi			Add1	Div1					Mult1			

Use the following blank tables for your answer.

Cycle 4 (8 marks)

Instr. status	Instructions (inst.)	Issue	Exec. Com-plete	Write result	RS status	RS Name	Busy	Op	Vj	Vk	Qj	Qk	A
	`MULT.S F8,F0,F2`	1	4			Add1	Y	A.S		R[F1]	M1		
	`ADD.S F2, F8, F1`	2				Mult1	Y	M.S	R[F0]	R[F2]			
	`DIV.S F3, F2, F1`	3				Mult2	N						
	`SUB.S F8, F2, F0`					Div1	Y	D.S		R[F1]	A1		
Reg. Status	Name	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
	Qi			Add1	Div1					Mult1			

Cycle 7 (8 marks)

Instr. status	Instructions (inst.)	Issue	Exec. Com-plete	Write result	RS status	RS Name	Busy	Op	Vj	Vk	Qj	Qk	A
	`MULT.S F8,F0,F2`	1	4	5		Add1	Y	A.S	Value	R[F1]			
	`ADD.S F2, F8, F1`	2	7			Mult1	N						
	`DIV.S F3, F2, F1`	3				Mult2	N						
	`SUB.S F8, F2, F0`					Div1	Y	D.S		R[F1]	A1		
Reg. Status	Name	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
	Qi			Add1	Div1								

Cycle 9 (8 marks)

Instr. status	Instructions (inst.)	Issue	Exec. Com- plete	Write result	RS status	RS Name	Busy	Op	Vj	Vk	Qj	Qk	A
	`MULT.S F8,F0,F2`	1	4	5		Add1	Y	S.S	R[F2]	R[F0]			
	`ADD.S F2, F8, F1`	2	7	8		Mult1	N						
	`DIV.S F3, F2, F1`	3	-5			Mult2	N						
	`SUB.S F8, F2, F0`	9				Div1	Y	D.S	value	R[F1]			
Reg. Status	Name	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
	Qi				Div1					Add1			