Computer Organization MIDTERM EXAM

2012.10.27.

name : **student number :**

Total point : 259

Chapter 1

1) Semiconductor DRAM and disk storage differ significantly. Describe the fundamental difference for each of the following: volatility, access time, and cost.[3]

2) A given application written in Java runs 15seconds on a desktop processor. A new Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler? Pick the right answer from the three choices below[3]
   ① $\frac{15\times0.6}{1.1} = 8.2sec$
   ② $15 \times 0.6 \times 1.1 = 9.9sec$
   ③ $\frac{15\times1.1}{0.6} = 27.5\text{sec}$

3) Consider the following performance measurements for a program:

Measurement	Computer A	Computer B
Instruction count	10 billion	8 billion
clock rate	4GHz	4GHz
CPI	1.0	1.1

   a. Which computer has the higher MIPS rating?[3]
   b. Which computer is faster?[3]

4) Assume that a design team is considering enhancing a machine by adding MMX (multimedia extension instruction) hardware to a processor. When a computation is run in MMX mode on the MMX hardware, it is 10 times faster than the normal mode of execution. Call the percentage of time that could be spent using the MMX mode the percentage of media enhancement. [15]
   (a) What percentage of media enhancement is needed to achieve an overall speedup of 2?
   (b) What percentage of the run-time is spent in MMX mode if a speedup of 2 is achieved? (Hint: You will need to calculate the new overall time.)
   (c) What percentage of the media enhancement is needed to achieve one-half the maximum speedup attainable from using the MMX mode?

5) (Exercise) Another pitfall is expecting to improve the overall performance of a computer by improving only one aspect of the computer. This might be true, but not always. Consider a computer running programs with CPU times shown in the following table.[20]

	FP instr.	INT instr.	L/S instr.	Branch instr.	Total time
a.	35 s	85 s	50 s	30 s	200 s
b.	50 s	80 s	50 s	30 s	210 s

   a. By how much is the total time reduced if the time for FP operations is reduced by 20%?
   b. By how much is the time for INT operations reduced if the total time is reduced by 20%?
   c. Can the total time can be reduced by 20% by reducing only the time for branch instructions?

   The following table shows the instruction type breakdown per processor of a given application executed in different numbers of processors.

	# Processors	FP instr.	INT instr.	L/S instr.	Branch Instr.	CPI (FP)	CPI (INT)	CPI (L/S)	CPI (Branch)
a.	1	560×10^6	2000×10^6	1280×10^6	256×10^6	1	1	4	2
b.	8	80×10^6	240×10^6	160×10^6	32×10^6	1	1	4	2

Assume that each processor has a 2GHz clock rate.

d. By how much must we improve the CPI of FP instructions if we want the program to run two times faster?
e. By how much must we improve the CPI of L/S instructions if we want the program to run two times faster?
f. By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 40% and the CPI of L/S and branch is reduced by 30%?

Chapter 2

6) What MIPS instruction does this represent? Choose from one of the four options below.[2]

op	rs	rt	rd	shamt	funct
0	8	9	10	0	34

① add $s0, $s1, $s2
② add $s2, $s0, $s1
③ add $s2, $s1, $s0
④ sub $s2, $s0, $s1

7) Which operations can isolate a field in a word? [2] why?[2]

① AND
② A shift left followed by a shift right

8) C has many statements for decisions and loops while MIPS has few. Which of the following do or do not explain this imbalance? [2] Why?[2]

① More decision statements make code easier to read and understand.
② Fewer decision statements simplify the task of the underlying layer that is responsible for execution.
③ More decision statements mean fewer lines of code, which generally reduces coding time.
④ More decision statements mean fewer lines of code, which generally results in the execution of fewer operations.

9) Why does C provide two sets of operators for AND (& and &&) and two sets of operators for OR (| and ||) while MIPS doesn't?[3]

① Logical operations AND and OR implement & and | while conditional branches implement && and ||.
② The previous statement has it backwards: && and || correspond to logical operations while & and | map to conditional branches.
③ They are redundant and mean the same thing: && and || are simply inherited from the programming language B, the predecessor of C.

10) What is the range of addresses for conditional branches in MIPS (K = 1024)?[3]

① Addresses between 0 and 64K – 1
② Addresses between 0 and 256K – 1
③ Addresses up to about 32K before the branch to about 32K after
④ Addresses up to about 128K before the branch to about 128K after

11) What is the range of addresses for jump and jump and link in MIPS (M = 1024K)?[3]

① Addresses between 0 and 64M – 1
② Addresses between 0 and 256M – 1
③ Addresses up to about 32M before the branch to about 32M after
④ Addresses up to about 128M before the branch to about 128M after
⑤ Anywhere within a block of 64M addresses where the PC supplies the upper 6 bits

⑥ Anywhere within a block of 256M addresses where the PC supplies the upper 4 bits

12) What is the MIPS assembly language instruction corresponding to the machine instruction with the value 0000 0000hex?[3]

① j
② R-format
③ addi
④ sll
⑤ mfc0
⑥ Undefined opcode: there is no legal instruction that corresponds to 0.

13) When do you use primitives like load linked and store conditional?[3]

① When cooperating threads of a parallel program need to synchronize to get proper behaviour for reading and writing shared data
② When cooperating processes on a uniprocessor need to synchronize for reading and writing shared data

14) Which of the advantages of an interpreter over a translator do you think was most important for the designers of Java?[3]

① Ease of writing an interpreter
② Better error messages
③ Smaller object code
④ Machine independence

15) (Exercise) In MIPS assembly, write an assembly language version of the following C code segment:

```
for (i = 0; i < 98; i ++) {
    C[i] = A[i + 1] - A[i] * B[i + 2]
}
```

Arrays A, B and C start at memory location *A000hex*, *B000hex* and *C000hex* respectively. Try to reduce the total number of instructions and the number of expensive instructions such as multiplies.? [15]

16) (Exercise) This exercise deals with recursive procedure calls. For the following problems, the table has an assembly code fragment that computes the factorial of a number. However, the entries in the table have errors, and you will be asked to fix these errors.[20]

a.

```
FACT:   addi  $sp, $sp, -8
        sw    $ra, 4($sp)
        sw    $a0, 0($sp)
        slti  $t0, $a0, 1
        beq   $t0, $0, L1
        addi  $v0, $0, 1
        addi  $sp, $sp, 8
        jr    $ra

L1:     addi  $a0, $a0, -1
        jal   FACT
        lw    $a0, 4($sp)
        lw    $ra, 0($sp)
        addi  $sp, $sp, 8
        mul   $v0, $a0, $v0
        jr    $ra
```

b.

```
FACT:   addi  $sp, $sp, -8
        sw    $ra, 4($sp)
        sw    $a0, 0($sp)
        slti  $t0, $a0, 1
        beq   $t0, $0, L1
        addi  $v0, $0, 1
        addi  $sp, $sp, 8
        jr    $ra

L1:     addi  $t0, $t0, -1
        jal   FACT
        lw    $a0, 4($sp)
        lw    $ra, 0($sp)
        addi  $sp, $sp, 8
        mul   $v0, $a0, $v0
        jr    $ra
```

a. The MIPS assembly program above computes the factorial of a given input. The integer input is passed through register $a0, and the result is returned in register $v0. In the assembly code, there are a few errors. Correct the MIPS errors.

b. For the recursive factorial MIPS program above, assume that the input is 4. Rewrite the factorial program to operate in a nonrecursive manner. Restrict your register usage to registers $s0 - $s7. What is the total number of instructions?

c. Show the contents of the stack after each function call, assuming that the input is 4.

For the following problems, the table has an assembly code fragment that computes a Fibonacci number. However, the entries in the table have errors, and you will be asked to fix these errors.

a.

```
FIB:  addi  $sp,$sp, -12
      sw    $ra, 0($sp)
      sw    $s1, 4($sp)
      sw    $a0, 8($sp)
      slti  $t0, $a0, 1
      beq   $t0, $0, L1
      addi  $v0,$a0, $0
      j     EXIT

L1:   addi  $a0,$a0, -1
      jal   FIB
      addi  $s1,$v0, $0
      addi  $a0,$a0, -1
      jal   FIB
      add   $v0, $v0, $s1

EXIT: lw    $ra, 0($sp)
      lw    $a0, 8($sp)
      lw    $s1, 4($sp)
      addi  $sp, $sp, 12
      jr    $ra
```

b.

```
FIB:  addi  $sp,$sp, -12
      sw    $ra, 0($sp)
      sw    $s1, 4($sp)
      sw    $a0, 8($sp)
      slti  $t0, $a0, 1
      beq   $t0, $0, L1
      addi  $v0,$a0, $0
      j     EXIT

L1:   addi  $a0,$a0, -1
      jal   FIB
      addi  $s1,$v0, $0
      addi  $a0,$a0, -1
      jal   FIB
      add   $v0, $v0, $s1

EXIT: lw    $ra, 0($sp)
      lw    $a0, 8($sp)
      lw    $s1, 4($sp)
      addi  $sp, $sp, 12
      jr    $ra
```

d. The MIPS assembly program above computes the Fibonacci of a given input. The integer input is passed through register $a0, and the result is returned in register $v0. In the assembly code, there are a few errors. Correct the MIPS errors.

e. For the recursive Fibonacci MIPS program above, assume that the input is 4. Rewrite the Fibonacci program to operate in a nonrecursive manner. Restrict your register usage to registers $s0 - $s7. What is the total number of instructions used to execute your solution from 2-16-b) versus the recursive version of the factorial program?

f. Show the contents of the stack after each function call, assuming that the input is 4.

17) (Exercise) The table below contains the link-level details of two different procedures. In this exercise, you will be taking the place of the linker.[20]

a.

	Procedure A				Procedure B			
Text Segment	Address	Instruction		Text Segment	Address	Instruction	
	0	lw $a0, 0($gp)			0	sw $a1, 0($gp)	
	4	jal 0			4	jal 0	
	...	...			...	...	
Data Segment	0	(X)		Data Segment	0	(Y)	
	...	...			...	...	
Relocation Info	Address	Instruction Type	Dependency	Relocation Info	Address	Instruction Type	Dependency
	0	lw	X		0	sw	Y
	4	jal	B		4	jal	A
Symbol Table	Address	Symbol		Symbol Table	Address	Symbol	
	–	X			–	Y	
	–	B			–	A	

b.

	Procedure A				Procedure B			
Text Segment	Address	Instruction		Text Segment	Address	Instruction	
	0	lui $at, 0			0	sw $a0, 0($gp)	
	4	ori $a0, $at, 0			4	jmp 0	
	8	jal 0			...	...	
	...	...			0x180	jr $ra	
					...	...	
Data Segment	0	(X)		Data Segment	0	(Y)	
	...	...			...	...	
Relocation Info	Address	Instruction Type	Dependency	Relocation Info	Address	Instruction Type	Dependency
	0	lui	X		0	sw	Y
	4	ori	X		4	jmp	FOO
	8	jal	B				
Symbol Table	Address	Symbol		Symbol Table	Address	Symbol	
	–	X			–	Y	
	–	B			0x180	FOO	

a. Link the object files above to form the executable file header. Assume that Procedure A has a text size of 0x140, data size of 0x40 and Procedure B has a text size of 0x300 and 0x50. Also assume the memory allocation strategy as shown in Figure 2-17).

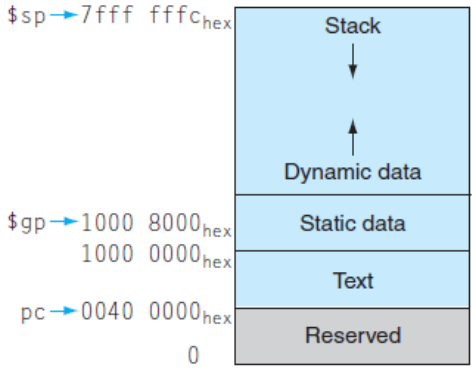


Figure 2-17)

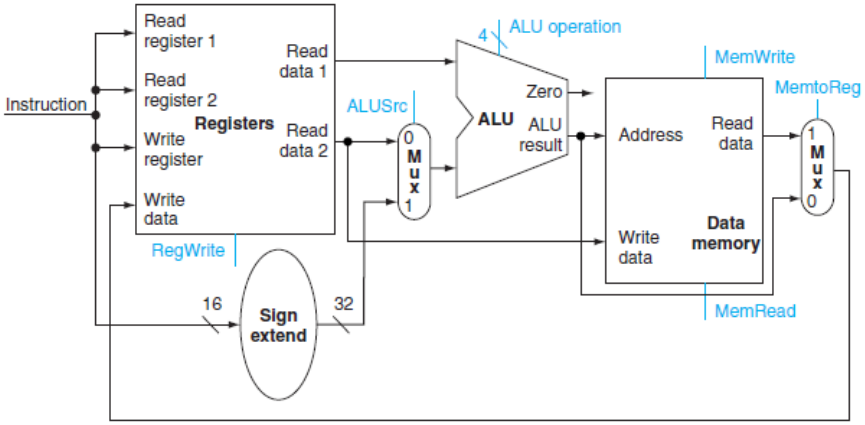
b. What limitations, if any, are there on the size of an executable?

c. Given your understanding of the limitations of branch and jump instructions, why might an assembler have problems directly implementing branch and jump instructions in an object file?

Chapter 4

18) (True or false): Because the register file is both read and written on the same clock cycle, any MIPS datapath using edge-triggered writes must have more than one copy of the register file.[3]

19) Which of the following is correct for a load instruction?[3]



① MemtoReg should be set to cause the data from memory to be sent to the register file.

② MemtoReg should be set to cause the correct register destination to be sent to the register file.

③ We do not care about the setting of MemtoReg for loads.

20) Look at the control signal in bellow figure. Can any control signal in the figure be replaced by the inverse of another? (Hint: Take into account the don't cares.) If so, can you use one signal for the other without adding an inverter? [3] why?[2]

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

21) For each code sequence below, state whether it must stall, can avoid stalls using only forwarding, or can execute without stalling or forwarding.[3]

Sequence 1	Sequence 2	Sequence 3

lw $t0,0($t0)
add $t1,$t0,$t0
	add $t1,$t0,$t0
	addi $t2,$t0,#5
	addi $t4,$t1,#5
		addi $t1,$t0,#1
		addi $t2,$t0,#2
		addi $t3,$t0,#2
		addi $t3,$t0,#4
		addi $t5,$t0,#5

22) A group of students have been debating the efficiency of the five-stage pipeline when one student pointed out that not all instructions are active in every stage of the pipeline. After deciding to ignore the effects of hazards, they made the following five statements. Which ones are correct?[3]

① Allowing jumps, branches, and ALU instructions to take fewer stages than the five required by the load instruction will increase pipeline performance under all circumstances.

② Trying to allow some instructions to take fewer cycles does not help, since the throughput is determined by the clock cycle; the number of pipe stages per instruction affects latency, not throughput.

③ Allowing jumps, branches, and ALU operations to take fewer cycles only helps when no loads or stores are in the pipeline, so the benefits are small.

④ You cannot make ALU instructions take fewer cycles because of the writeback of the result, but branches and jumps can take fewer cycles, so there is some opportunity for improvement.

⑤ Instead of trying to make instructions take fewer cycles, we should explore making the pipeline longer, so that instructions take more cycles, but the cycles are shorter. This could improve performance.

23) Consider three branch prediction schemes: branch not taken, predict taken, and dynamic prediction. Assume that they all have zero penalty when they predict correctly and 2 cycles when they are wrong. Assume that the average predict accuracy of the dynamic predictor is 90%. Which predictor is the best choice for the following branches?[3]

① A branch that is taken with 5% frequency

② A branch that is taken with 95% frequency

③ A branch that is taken with 70% frequency

24) Which exception should be recognized first in this sequence?[3]

① add $1, $2, $1 # arithmetic overflow

② XXX $1, $2, $1 # undefined instruction

③ sub $1, $2, $1 # hardware error

25) State whether the following techniques or components are associated primarily with a software- or hardware-based approach to exploiting ILP. In some cases, the answer may be both.[4]

① Branch prediction

② Multiple issue

③ VLIW

④ Superscalar

⑤ Dynamic scheduling

⑥ Out-of-order execution

⑦ Speculation

⑧ Reorder buffer

⑨ Register renaming

26) In this question we will explore how bypassing affects program execution performance. To begin consider the standard MIPS 5 stage pipeline. For your reference, refer to the figure below. For this question, we will use the following code to evaluate the pipeline's performance: [20]

```
1 add $t2, $s1, $sp
2 lw $t1, $t2, 0
3 addi $t2, $t1, 7
4 add $t1, $s2, $sp
5 lw $t1, $t1, 0
6 addi $t1, $t1, 9
7 sub $t1, $t1, $t2
```

(a) What is the load-use latency for the standard MIPS 5-stage pipeline?

(b) Once again, using the standard MIPS pipeline, identify whether the value for each register operand is coming from the bypass or from the register file. For clarity, please write REG or BYPASS in each box.

Instruction	Src Operand 1	Src Operand 2
1		
2		N/A
3		N/A
4		
5		N/A
6		N/A
7		

(c) How many cycles will the program take to execute on the standard MIPS pipeline?

(d) Assume, due to circuit constraints, that the bypass wire from the memory stage back to the execute stage is omitted from the pipeline. What is the load-use latency for this modified pipeline?

(e) Identify whether the value for each register operand is coming from the bypass or from the register file for the modified pipeline. For clarity, please write REG or BYPASS in each box.

Instruction	Src Operand 1	Src Operand 2
1		
2		N/A
3		N/A
4		
5		N/A
6		N/A
7		

(f) How long does the program take to execute on the modified pipeline?

27) (Exercise) This problem is intended to help you understand the relationship between delay slots, control hazards, and branch execution in a pipelined processor. In this problem, we assume that the following MIPS code is executed on a pipelined processor with a five-stage pipeline, full forwarding, and a predict-taken branch predictor:[20]

a.
```
Label1: lw  $1,40($6)
        beq $2,$3,Label2 ; Taken
        add $1,$6,$4
Label2: beq $1,$2,Label1 ; Not taken
        sw  $2,20($4)
        and $1,$1,$4
```

b.
```
        add $1,$5,$3
Label1: sw  $1,0($2)
        add $2,$2,$3
        beq $2,$4,Label1 ; Not taken
        add $5,$5,$1
        sw  $1,0($2)
```

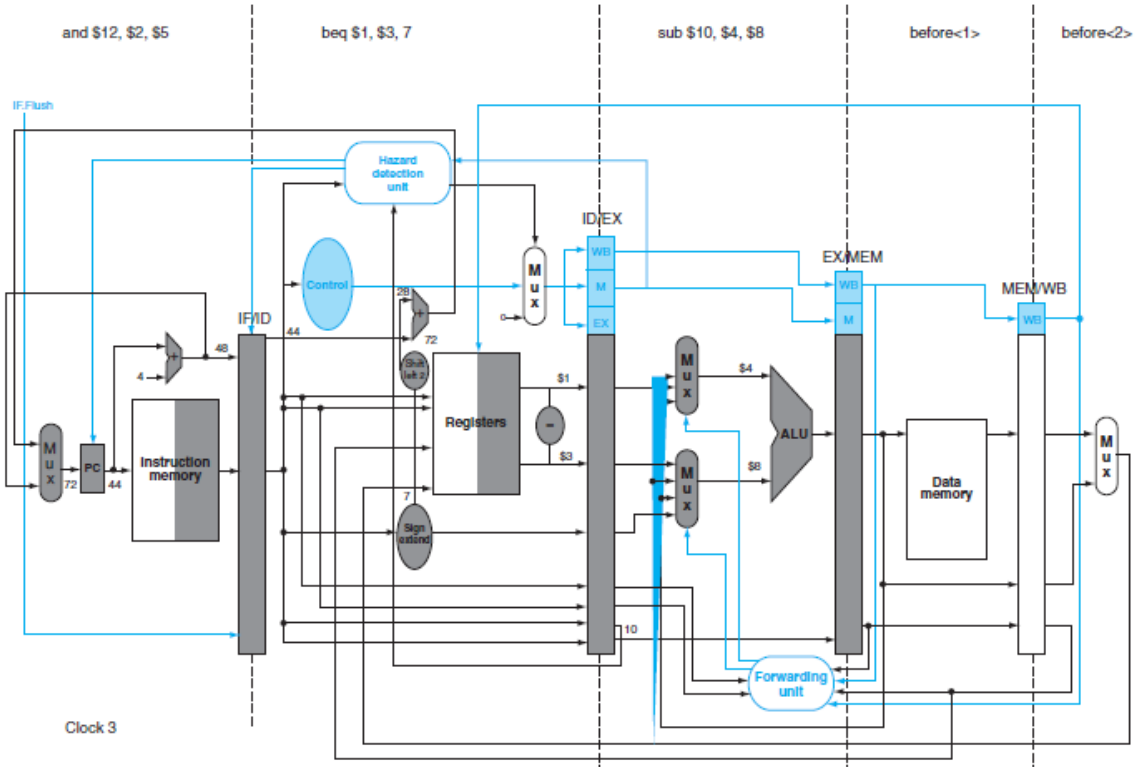


Figure 4-27)

a. Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.
b. Repeat 4-27-a), but assume that delay slots are used. In the given code, the instruction that follows the branch is now the delay slot instruction for that branch.
c. One way to move the branch resolution one stage earlier is to not need an ALU operation in conditional branches. The branch instructions would be “bez RD,Label” and “bnez RD,Label”, and it would branch if the register has and does not have a 0 value, respectively. Change this code to use these branch instruction instead of ‘beq’. You can assume that register $8 is available for you to use as a temporary register, and that a ‘seq’(set if equal) R-type instruction can be used.
d. Using the first branch instruction in the given code as an example, describe the hazard detection logic needed to support branch execution in the ID stage as shown in Figure 4-27). Which type of hazard is this new logic supposed to detect?
e. For the given code, what is the speed-up achieved by moving branch execution into the ID stage? Explain your answer. In your speed-up calculation, assume that the additional comparison in the ID stage does not affect clock cycle time.
f. Using the first branch instruction in the given code as an example, describe the forwarding support that must be added to support branch execution in the ID stage. Compare the complexity of this new forwarding unit to the complexity of the existing forwarding unit in Figure 4-27).

Chapter 5

28) Which of the following statements are generally true?[3]

① Caches take advantage of temporal locality.
② On a read, the value returned depends on which blocks are in the cache.
③ Most of the cost of the memory hierarchy is at the highest level.
④ Most of the capacity of the memory hierarchy is at the lowest level.

29) The speed of the memory system affects the designer's decision on the size of the cache block. Which of the following cache designer guidelines are generally valid?[3]

① The shorter the memory latency, the smaller the cache block.

② The shorter the memory latency, the larger the cache block.

③ The higher the memory bandwidth, the smaller the cache block.

④ The higher the memory bandwidth, the larger the cache block.

30) Which of the following is generally true about a design with multiple levels of caches?[3]

① First-level caches are more concerned about hit time, and second-level caches are more concerned about miss rate.

② First-level caches are more concerned about miss rate, and second-level caches are more concerned about hit time.

31) (Exercise) In this problem, we will examine how replacement policies impact miss rate. Assume a two-way set-associative cache with four blocks. There is example (below table) as demonstrated below on the address sequence "0, 1, 2, 3, 4".[20]

Address of memory block accessed	Hit or miss	Evicted block	Contents of cache blocks after			
			Set0	Set0	Set1	Set1
0	Miss		Mem[0]			
1	Miss		Mem[0]		Mem[1]	
2	Miss		Mem[0]	Mem[2]	Mem[1]	
3	Miss		Mem[0]	Mem[2]	Mem[1]	Mem[3]
4	Miss	0	Mem[4]	Mem[2]	Mem[1]	Mem[3]
…						

The following table shows address sequences.

	Address sequence
a.	0, 2, 4, 0, 2, 4, 0, 2, 4
b,	0, 2, 4, 2, 0, 2, 4, 0, 2

a. Assuming an LRU replacement policy, how many hits does this address sequence exhibit?

b. Assuming an MRU(most recently used) replacement policy, how many hits does this address sequence exhibit?

c. Simulate a random replacement policy by flipping a coin. For example, "heads" means to evict the first block in a set and "tails" means to evict the second block in a set. How hits does this address sequence exhibit?

d. Which address should be evicted at each replacement to maximize the number of hits? How many hits does this address sequence exhibit if you follow this "optimal" policy?

e. Describe why it is difficult to implement a cache replacement policy that is optimal for all address sequences.

f. Assume you could make a decision upon each memory reference whether or not you want the requested address to be cached. What impact could this have on miss rate?

32) What are the two characteristics of program memory accesses that caches exploit? [5]

33) Describe three types of cache misses: cold misses, conflict misses and compulsory misses [15]

34) Design an 8-way set associative cache that has 16 blocks and 32 bytes per block. Assume a 32 bit address. Calculate the following: [15]

(a) How many bits are used for the byte offset?

(b) How many bits are used for the set (index) field?

(c) How many bits are used for the tag?