

# SOLUTION FOR

## CS 216 Exam 2 – Fall 2005

Name \_\_\_\_\_ Section \_\_\_\_\_

Email Address \_\_\_\_\_ Student ID # \_\_\_\_\_

### Pledge:

This exam is closed note, closed book, with the exception of the “IBCM Principles of Operation” and the “Code Examples”, and the “Tiny Guide to x86 Assembly”. You may view these on-line - at the computer in front of you off of the cs216 home page – you may NOT use printed copies you may have brought with you. You may NOT refer to any other notes/books/slides/old exams etc.

You will have an hour and forty-five minutes total to complete the exam. You may use calculators if needed (including the calculator on Windows).

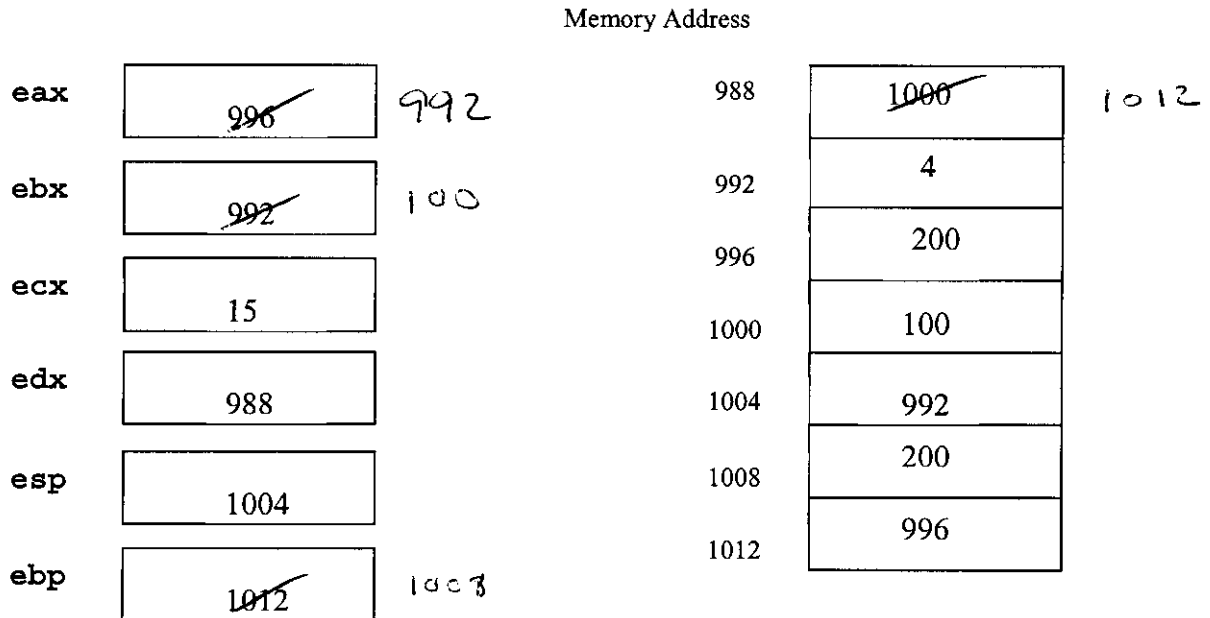
**It is an Honor Code violation to discuss this exam with ANYONE (including other students who have already taken the exam) until after noon, Thursday, Nov. 17, 2005.**

Good Luck!

	MAX	SCORE
TOTAL	72	

74

1. (6 points):



Modify the picture above to show what happens to the stack and the contents of the registers after the following instructions have been executed. Assume all values are decimal.

```
pop eax
mov ebx, [edx + 12]
mov [edx], ebp
sub ebp, 4
xor ecx, ecx
```

2. (6 points) In the worst case, <sup>(A)</sup> how long does it take to determine that an element is NOT in a hash table. Assume that the hash table has been implemented using separate chaining, where each bucket is a linked-list. <sup>(B)</sup> You must describe how you came up with your answer for any credit. Also, what would the worst-case be if each bucket is an AVL tree?

- (A) -  $\Theta(n)$  where  $n$  is number of items in table
- (B) - all items hash to same bucket
- (C) -  $\Theta(\lg n)$

3. (4 points) Give the Big-Oh for the average case time-complexity for the find operation for an AVL tree. What is it for a hash-table using separate chaining?

$\Theta(\lg n)$   
 $\Theta(1)$  or  $\Theta(\lambda)$  (the former assumes a  $\lambda$  value that's good)

4. (2 points) What load-factor does the textbook recommend for a hash-table when we use separate chaining? When we use open-addressing?

$\lambda = 1$        $\lambda = 0.5$

5. (6 points) Draw the contents of the hash table in the boxes below given the following conditions. Note that the calculator in Windows will do modulo arithmetic.  
 Choose Start->run, and type calc. Select view->scientific.  
**No partial credit – check your answers carefully!**

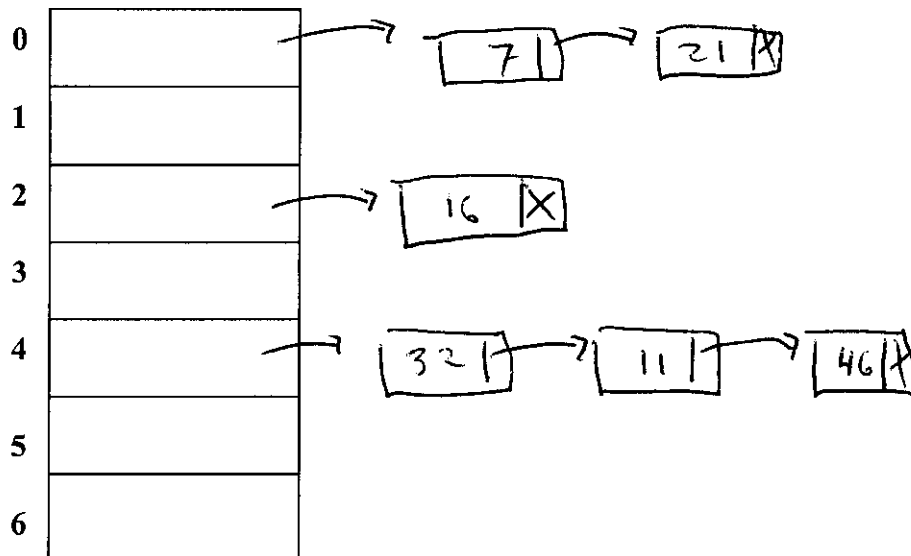
The size of the hash table is 7.

Use **separate chaining**, where each bucket is an **unordered linked list**.

The hash function used is  $H(k) = k \bmod \text{tablesize}$

What values will be in the hash table after the following sequence of insertions (draw in boxes below):

16, 7, 32, 11, 46, 21



Note: ↑  
array of pointers

↑  
list nodes

6. (5 points) Draw the contents of the hash table in the boxes below given the following conditions. Note that the calculator in Windows will do modulo arithmetic. Choose Start->run and type calc. Select view->scientific

**No partial credit – check your answers carefully!**

The size of the hash table is 8.

Open addressing and **linear probing** is used to resolve collisions.

The hash function used is  $H(k) = (k*2) \bmod \text{tablesize}$

What values will be in the hash table after the following sequence of insertions (draw in boxes below)?

7, 16, 22, 30, 10

0	16
1	
2	
3	
4	22
5	30
6	7
7	10



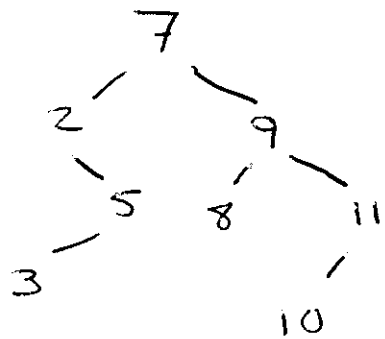
~~Note:~~

Note:

array of  
int

7. (4 points) Draw the binary search tree created by inserting these values in this order:

7 9 11 2 8 10 5 3



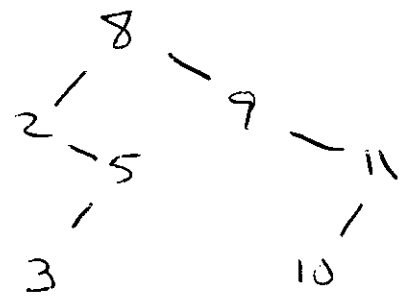
8. (2 points) Give a pre-order traversal of your tree shown above:

7 2 5 3 9 8 11 10

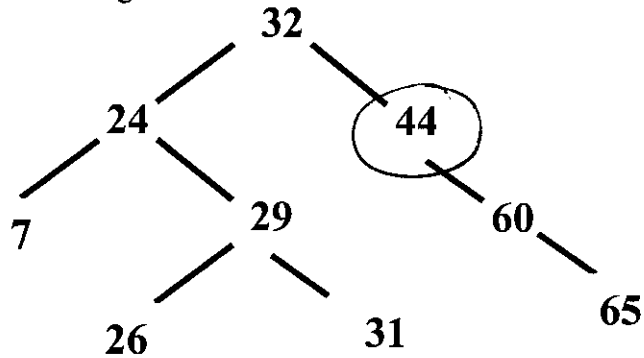
9. (2 points) Give an in-order traversal of your tree shown above:

2 3 5 7 8 9 10 11

10. (4 points) Delete the root of the tree shown above using one of the methods described in class. Draw the new tree here:



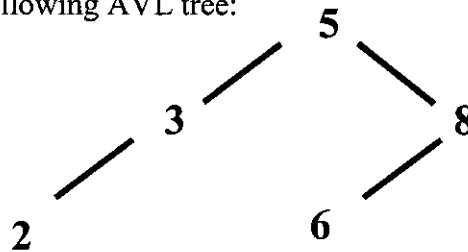
11. (3 points) Given the following tree:



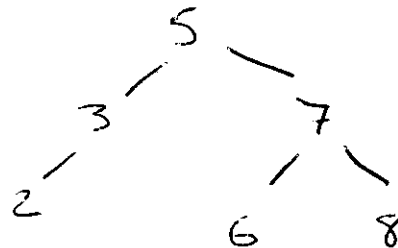
Is it an AVL tree? YES or NO (If NO, circle the node(s) where the AVL property is violated.)  
Why or why not (must answer for any credit)?

NO depth of left subtree and right subtree of (44) differ by more than 1

12. (4 points) Given the following AVL tree:



Insert the value 7 into the AVL tree above, doing any necessary rotations to maintain the AVL property.



13. (10 pts) Write x86 assembly code that implements the following pseudo-code. You must comment your code.

```
x = 1;
y = 4;
while ( y >= x ) {
    --y;
}
```

NOTES

There are many ways to code this, but they all have much in common. We looked for the following things:

- a) Declare variables with initial values  
--- or ---  
move values into memory locations
- b) Have a label for the top of the loop
- c) Compare and jump on the result  
Note: If you did `cmp y,x` you needed to use `jl`  
If you did `cmp x,y` you needed to use `jg`
- d) Decrement the register containing `y` (or the memory location for `y`)
- e) Jump back to the top of the loop
- f) Save the register containing `y` back to `y`'s memory location

We took off a point or two if you didn't store `x` and `y` in memory but only used registers.

In case it is useful to you, the following table provides the operation code definitions for IBCM. It is the same as the table in class handouts.

OP	Mnem	Note	OP	Mnem	Note
0	halt	halt!	8	or	logical OR mem into accum
1	io	bit 4 I/O, bit 5 hex/ascii	9	xor	logical XOR mem into accum
2	shift	bit 4 shift/rotate, bit 5 left/right	A	not	logical complement of accum
3	load	load accum from mem	B	nop	no operation, do nothing
4	store	store accum to mem	C	jmp	unconditional jump
5	add	add mem to accum	D	jmpe	jump to addr if accum is 0
6	sub	subtract mem from accum	E	jmpL	jump to addr if accum < 0
7	and	logical AND mem into accum	F	brl	jump to addr; set accum to value of PC just before jump

14. (16 points) Write an IBCM program that implements the following pseudo-code. Your answer should be machine code (encoded). First write symbolic IBCM instructions for significant partial credit. Then encode the IBCM instructions for full credit. For full credit on this question, encoded instructions and symbolic assembly is what is required. Comments will help us assign you partial credit in case your solution is not perfect, but are not required for full credit. You can assume that  $x$  and  $y$  will be integers  $\geq 0$ . Please clearly indicate your final answer.

Note: the pseudo-code below is very similar to that in the previous x86 question!

```

read x;
read y;
while (y >= x) {
    print y;
    --y;
}
halt

```

~~Write your~~

SOLUTION

mem	loc	label	op	addr	comments
c006	00		jmp	start	skip variables
0000	01	x	dw	0	
0000	02	y	dw	0	
0001	03	one	dw	1	
0000	04				
0000	05				
1000	06	start	readH		
4001	07		store	x	
1000	08		readH		
4002	09		store	y	
3002	0A	loop	load	y	
6001	0B		sub	x	
E012	0C		jmpL	xit	
3002	0D		load	y	
1800	0E		printH		
6003	0F		sub	one	
4002	10		store	y	
C00A	11		jmp	loop	
0000	12	xit	halt		