University of Virginia Computer Science
CS216: Program and Data Representation, Spring 2006 20 April 2006

Exam 2

Out: 20 April 2006
Due: Monday, 24 April, 11:01AM

Name: ____

Scores

1	2	3	4	5	6	7	8	9	Total
10	10	10	10	10	5	10	10	10	85

Directions

Work alone. You may not discuss these problems or anything related to the material covered by this exam with anyone except for the course staff between receiving this exam and class Monday.

Closed web. You may not search the web to attempt to find answers to the questions on this exam. You may you web pages linked from the CS216 web site, but may not do web searches to attempt to find specific answers.

Open other resrouces. You may use any books you want, lecture notes and slides, your notes, and problem sets. If you use anything other than the course books and notes, cite what you used. You may not use other people.

Open tools. You *may* run any program you want, including a Python interpreter, C compiler, Java compilers, Java VM, and x86 assembler for this exam. You are not expected to need to do this, and will not lose points for minor syntactic mistakes.

Answer well. Write your answers on this exam. You should not need more space than is provided to write good answers, but if you want more space you may attach extra sheets. If you do, make sure the answers are clearly marked.

This exam has 10 questions, the last of which is non-credit. The questions are not necessarily in order of increasing difficulty, so if you get stuck on one question you should continue on to the next question. There is no time limit on this exam, but it should not take a well-prepared student more than a few hours to complete.

Full credit depends on the clarity and elegance of your answer, not just correctness. Your answers should be as short and simple as possible, but not simpler.

Huffman Encoding

1. (10) Consider the following frequency distribution:

Symbol:	A	B	C	D	E	F	G
Count:	5	3	2	3	6	2	4

How many different *optimal* prefix encodings are there for the given frequency distribution? Your answer should include a clear explanation of why it is correct.

Number Representations

2. (10) In Class 16, we saw that the floating point imprecision in representing 0.1 led to an error of 0.0034 seconds per hour in the Patriot missle time calculations. What clock tick unit would maximize the error accumulated per hour? What is the error?

Memory Management

3. (10) Explain (a) why the C program below has a memory leak and (b) how to fix it.

```
# include
# include
# include

char *copyString (char *s)
{
  char *res = (char *) malloc (sizeof (char) * strlen (s));
  strcpy (res, s);
  return res;
}

int main (int argc, char **argv)
{
  char *a = "alpha";
  char *b = "beta";

  while (*a != *b) {
    b = copyString (b + 1);
  }

  printf ("The strings are: %s / %s\n", a, b);
  exit (0);
}
```

4. (10) Explain two reasons why it is easier to write a garbage collector for Python than it is to write a garbage collector for C?

5. (10) Here is the JVML code for a Java method:

```
Method int func(int, int)
   0 iload_0
   1 istore_2
   2 iload_1
   3 istore_3
   4 iload_2
   5 iload_3
   6 iadd
   7 istore 4
   9 iload_2
  10 iload 4
  12 if_icmple 18
  15 iinc 4 1
  18 iload 4
  20 ireturn
```

Write JVML code for a method with exactly the same behavior with as few instructions as possible. Be careful to make sure the result from your new function will always match the result from the original function on all possible inputs.

Assembly Programming

For each of the next three questions, answer whether or not the two shown assembly code fragements have equivalent behavior. *Equivalent behavior* is defined as if the values in all general purpose registers (we do not consider the flag registers), the stack, and all of memory are the same before entering the fragment, they are always the same after exiting the fragment. If the two fragements have the same behavior, explain what that behavior is. If they have different behavior, illustrate the difference by showing an initial state for which the two fragments produce different final states.

6. (5)

Fragment A

```
mov eax, ebx
```

Fragment B

```
push ecx
mov ebx, ecx
mov eax, ecx
mov cx, bx
pop ecx
```

7. (10) For this question, assume the called function `_func` correctly follows the C calling convention.

Fragment A

```
push 216
push 202
call _func
add esp, 8
```

Fragment B

```
push eax
push 216
push 202
call _func
add esp, 8
pop eax
```

8. (10) Do the two functions have equivalent behavior? (Assume all callers must correctly follow the C calling convention.)

```
_myFunc PROC
  ; Subroutine Prologue
  push ebp     ; Save the old base pointer value.
  mov ebp, esp ; Set the new base pointer value.
  sub esp, 4   ; Make room for one 4-byte local variable.
  push edi     ; Save the values of modified registers.
  push esi     ; (no need to save EBX, EBP, or ESP)

  ; Subroutine Body
  mov eax, [ebp+8]   ; Move parameter 1 into EAX
  mov esi, [ebp+12]  ; Move parameter 2 into ESI
  mov edi, [ebp+16]  ; Move parameter 3 into EDI

  mov [ebp-4], edi   ; Move EDI into local variable
  add [ebp-4], esi   ; Add ESI into local variable
  add eax, [ebp-4]   ; Add local into EAX (result)

  ; Subroutine Epilogue
  pop esi      ; Recover register values
  pop  edi
  mov esp, ebp ; Deallocate local variables
  pop ebp      ; Restore the caller's ebp
  ret
_myFunc ENDP
END
```

Fragment A (Note: this is the example from the x86 Guide, with some of the comments shortened to save space)

```
_myFunc PROC
  sub esp, 4
  mov eax, [esp+8]
  mov ecx, [esp+12]
  mov edx, [esp+16]
  mov [esp], edx
  add [esp], ecx
  add eax, [esp]
  pop edx
  ret
_myFunc ENDP
END
```

Fragment B

9. (10) Consider modifying the x86 calling convention to put the return value on the top of the stack when a routine returns instead of using EAX to hold the return value. What are the advantages and disadvantages of this change?

10. (no credit) Which topics would you most prefer we cover in the remaining two classes (use "1" to indicate your most prefered topic, "2" for second most, etc.):

____ Improved Tree Data Structures

____ Partner Assignment Algorithms

____ Graph and Network Algorithms

____ Randomized Algorithms

____ .NET's virtual machine and how it differs from the Java VM

____ An instruction set very different from x86

____ Review

____ Other: ____

cs216-staff@cs.virginia.edu