Once-in-a-Semester-Special Richard Guo
Mock "Faux Examen" 07/16/08

Memory management/C:

Draw out and label the four parts of memory and mark how much is allocated in each section (except for code, whose size you don't know) at the point in execution in which the program uses the most memory. Assume a 32 bit system. (Hint: Think about which registers the function needs to preserve)

```
#include <stdio.h>
#define SHIFT_VAL 1

int g_count = 32;
char* str = "O woe is me, most unhappy day.";

unsigned int foo (int a)
{
        if (g_count == 0)
                return 0;
        g_count--;
        if (a < 0)
                return 1 + 2*foo(a <<= SHIFT_VAL);
        else
                return 2*foo(a*2);
}

int main()
{
        printf("%u", foo(1));
        return 0;
}
```

0 bytes for preprocessor defines
4 bytes int, 4 bytes char* in static
31 bytes for string in heap
foo loops through 32 times (4bytes * 32)
Doesn't need to save $a0 or $v0 b/c they're not needed after
Total = 31+8+128= 167 bytes

What is the function of foo? What will the function print out?
(reverses the bits of a and returns)   (2147483648, or 2^31)

Number Representation:
Order the bitwise representations of the following numbers as unsigned integers from largest to smallest (assume 32 bit space). You do not necessarily need to calculate the bits; just list the letters in order:

a. sign and magnitude representation of (-2147483647 + 12345) ***{0xFFFFCFC6}***
b. upper bits of the double precision floating point 2^(-523) ***{0x1f4...}***
c. 0x9c5a6821 ***{~2.25 trillion, negative → actual 2623170593}***
d. one's complement form of (-2147483647 + 12345) ***{0x80003039 = 0b10000000000000000011000000111010}***
e. two's complement form of (-2147483647 + 12345) ***{inv(LONG_MAX-12345) = 2^31-1 - 12345 → 0x8000303a}***
f. 1 Yobibyte / 1 Pebibyte represented as an integer ***{0x40000000}***

Ans: a > c > e > d > f > b

MIPS:

Be a compiler. Translate the following C into MIPS TAL:

```
int dot_product (pixel* pix1, pixel* pix2)
{
      int temp;
      pix1->r *= pix2->r;
      pix1->g *= pix2->g;
      pix1->b *= pix2->b;

      temp = (pix1->r + pix1->g + pix1->b) / 3;

      if (temp <= 255)
      {
            pix1->moy = temp;
            return 0;
      }
      return 1;
}
```

```
Assume that the following data type has been declared and translated for you:
typedef struct colour
{
      short r;
      short g;
      short b;
      char moy;
} pixel;
```

Ans:

```
dot_product:addi $v0, $0, 1
            lhu $t0, 0($a0) #r
            lhu $t1, 0($a1) #r2
            mult $t0, $t1
            mflo $t7
            sh $t7, 0($a0)

            lhu $t0, 2($a0) #g
            lhu $t1, 2($a1) #g2
            mult $t0, $t1
            mflo $t8
            sh $t8, 2($a0)

            lhu $t0, 4($a0) #b
            lhu $t1, 4($a1) #b2
            mult $t0, $t1
            mflo $t9
            sh $t9, 4($a0)

            addu $t7, $t7, $t8
            addu $t7, $t7, $t9
            addi $t0, $0, 3
            div $t7, $t0
            mflo $t7
            sltiu $t0, $t7, 256
            beq $t0, $0, end
            sb $t7, 6($a0)
            add $v0, $0, $0
end:        jr $ra
```

CALL:

* T/F: If a program (a single .c file with a main() statement) does not rely on external libraries or files, it does not need to undergo the linker step. As in, it can be loaded and executed immediately after the assembler step. ***{F – object file cannot be loaded even though it's machine code; certain things have not been added, e.g. table of local symbols, additional bits the loader needs have not been added}***

<u>The following 3 questions apply to C programs on computers without software emulation</u>

* The resulting code is portable between different operating systems (e.g. Solaris and Windows) (circle all that apply)...

before compilation
after compilation but before assembling
after assembling but before linking
after linking but before loading

Ans: All of the above. Code is portable all the way up to the binary level (look at Wine for example)

* The resulting code is portable between different hardware ISAs (e.g. x86, MIPS, SPARC, and PowerPC) (circle all that apply)...

before compilation
after compilation but before assembling
after assembling but before linking
after linking but before loading

Ans: Only compilation. Once compiled to machine code, it cannot be moved to another ISA. An exception is Apple's Rosetta, but that is a form of "software emulation."

* T/F: If we compile & assemble a program that uses dynamic libraries on Solaris, can we move that object file to a MS Windows computer and link with the Windows versions of those libraries? ***{F – ELF format vs MS proprietary PE format}***