Namn:____

Personnummer:____

Datorarkitektur, 2007

Tentamen 2007-03-09

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maximum score of 60 points plus 3 possible bonus points.
- The aproximate limits for grades on this exam are:
  - To pass (G or 3): 30 points.
  - For grade 4: 43 points.
  - For grade VG: 50 points.
  - For grade 5: 55 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like. Good luck!

Problem 1. (9 points):

Assume we are running code on a 6-bit machine using two's complement arithmetic for signed integers. A "short" integer is encoded using 3 bits. Fill in the empty boxes in the table below. The following definitions are used in the table:

```
short sy = -3;
int y = sy;
int x = -17;
unsigned ux = x;
```

Note: You need not fill in entries marked with "–".

Expression	Decimal Representation	Binary Representation
Zero	0	
–	-6	
–		`01 0010`
ux		
y		
$x >> 1$		
TMax		
$-$TMin		
TMin $+$ TMin		

Problem 2. (8 points):

The following procedure takes a single-precision floating point number in IEEE format and prints out information about what category of number it is. Fill in the missing code so that it performs this classification correctly.

```
void classify_float(float f)
{
  /* Unsigned value u has same bit pattern as f */
  unsigned u = *(unsigned *) &f;

  /* Split u into the different parts */
  int sign = (u >> 31) & 0x1;      // The sign bit

  int exp  = _______________;      // The exponent field

  int frac = _______________;      // The fraction field

  /* The remaining expressions can be written in terms of the
  values of sign, exp, and frac */

  if (______________________)
    printf("Plus or minus zero\");

  else if (______________________)
    printf("Nonzero, denormalized\");

  else if (______________________)
    printf("Plus or minus infinity\");

  else if (______________________)
    printf("NaN\");

  else if (______________________)
    printf("Greater than -1.0 and less than 1.0\");

  else if (______________________)
    printf("Less than or equal to -1.0\");

  else
    printf("Greater than or equal to 1.0\");
}
```

Problem 3. (8 points):

Consider the following IA32 code for a procedure `foo()`:

```
  foo:
      pushl %ebp
      movl %esp,%ebp
      movl 8(%ebp),%ecx
      movl 16(%ebp),%edx
      movl 12(%ebp),%eax
      decl %eax
      js .L3
 .L7:
      cmpl %edx,(%ecx,%eax,4)
      jne .L3
      decl %eax
      jns .L7
 .L3:
      movl %ebp,%esp
      popl %ebp
      ret
```

Based on the assembly code above, fill in the blanks below in its corresponding C source code. (Note: you may only use symbolic variables a, n, val, and i from the source code in your expressions below—do *not* use register names.)

```
int foo(int *a, int n, int val) {
  int i;

  for (i = _________; ____________________________ ; i =___________) {
     ;
  }
  return i;
}
```

Problem 4. (7 points):

Consider the following code fragment containing the incomplete definition of a data type `matrix_entry` with 4 fields.

```
struct matrix_entry{

  ____ a;

  ____ b;

  int  c;

  ____ d;

};

struct matrix_entry matrix[2][5];

int return_entry(int i, int j){
   return matrix[i][j].c;
}
```

Complete the above definition of `matrix_entry` so that the following assembly code could be generated from it on a Linux/x86 machine:

```
return_entry:
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%eax
        leal (%eax,%eax,4),%eax
        addl 12(%ebp),%eax
        sall $4,%eax
        movl matrix+4(%eax),%eax
        movl %ebp,%esp
        popl %ebp
        ret
```

Notes

- Note that there are multiple correct answers.
- Choose your answers from the following types, assuming the following sizes and alignments:

Type	Size (bytes)	Alignment (bytes)
char	1	1
short	2	2
int	4	4
double	8	4

Problem 5. (8 points):

The following problem concerns the following, low-quality code:

```
void foo(int x)
{
  int a[3];
  char buf[4];
  a[0] = 0xF0F1F2F3;
  a[1] = x;
  gets(buf);
  printf("a[0] = 0x%x, a[1] = 0x%x, buf = %s\n", a[0], a[1], buf);
}
```

In a program containing this code, procedure `foo` has the following disassembled form on an IA32 machine:

```
080485d0 <foo>:
80485d0:       55                pushl  %ebp
80485d1:       89 e5             movl   %esp,%ebp
80485d3:       83 ec 10          subl   $0x10,%esp
80485d6:       53                pushl  %ebx
80485d7:       8b 45 08          movl   0x8(%ebp),%eax
80485da:       c7 45 f4 f3 f2    movl   $0xf0f1f2f3,0xfffffff4(%ebp)
80485df:       f1 f0
80485e1:       89 45 f8          movl   %eax,0xfffffff8(%ebp)
80485e4:       8d 5d f0          leal   0xfffffff0(%ebp),%ebx
80485e7:       53                pushl  %ebx
80485e8:       e8 b7 fe ff ff    call   80484a4 <_init+0x54>  # gets
80485ed:       53                pushl  %ebx
80485ee:       8b 45 f8          movl   0xfffffff8(%ebp),%eax
80485f1:       50                pushl  %eax
80485f2:       8b 45 f4          movl   0xfffffff4(%ebp),%eax
80485f5:       50                pushl  %eax
80485f6:       68 ec 90 04 08    pushl  $0x80490ec
80485fb:       e8 94 fe ff ff    call   8048494 <_init+0x44>  # printf
8048600:       8b 5d ec          movl   0xffffffec(%ebp),%ebx
8048603:       89 ec             movl   %ebp,%esp
8048605:       5d                popl   %ebp
8048606:       c3                ret
8048607:       90                nop
```

For the following questions, recall that:

- `gets` is a standard C library routine.
- IA32 machines are little-endian.
- C strings are null-terminated (i.e., terminated by a character with value 0x00).
- Characters ‘`0`’ through ‘`9`’ have ASCII codes `0x30` through `0x39`.

Consider the case where procedure `foo` is called with argument `x` equal to `0xE3E2E1E0`, and we type "`123456789`" in response to `gets`.

A. Fill in the following table indicating which program values are/are not corrupted by the response from `gets`, i.e., their values were altered by some action within the call to `gets`.

Program Value	Corrupted? (Y/N)
`a[0]`	
`a[1]`	
`a[2]`	
`x`	
Saved value of register `%ebp`	
Saved value of register `%ebx`	

B. What will the `printf` function print for the following:

- `a[0]` (hexadecimal): ______________________________
- `a[1]` (hexadecimal): ______________________________
- `buf` (ASCII): ______________________________

Problem 6. (8 points):

The following problem concerns optimizing a procedure for maximum performance on an Intel Pentium III. Recall the following performance characteristics of the functional units for this machine:

Operation	Latency	Issue Time
Integer Add	1	1
Integer Multiply	4	1
Integer Divide	36	36
Floating Point Add	3	1
Floating Point Multiply	5	2
Floating Point Divide	38	38
Load or Store (Cache Hit)	1	1

You've just joined a programming team that is trying to develop the world's fastest factorial routine. Starting with recursive factorial, they've converted the code to use iteration:

```
int fact(int n)
{
  int i;
  int result = 1;

  for (i = n; i > 0; i--)
    result = result * i;

  return result;
}
```

By doing so, they have reduced the number of cycles per element (CPE) for the function from around 63 to around 4 (really!). Still, they would like to do better.

One of the programmers heard about loop unrolling. He generated the following code:

```
int fact_u2(int n)
{
  int i;
  int result = 1;

  for (i = n; i > 0; i-=2) {
    result = (result * i) * (i-1);
  }

  return result;
}
```

Unfortunately, the team has discovered that this code returns 0 for some values of argument `n`.

A. For what values of `n` will `fact_u2` and `fact` return different values?

B. Show how to fix `fact_u2` so that its behavior is identical to `fact`. [Hint: there is a special trick for this procedure that involves modifying just a single character.]

C. Benchmarking `fact_u2` shows no improvement in performance. How would you explain that?

D. You modify the line inside the loop to read:

```
result = result * (i * (i-1));
```

To everyone's astonishment, the measured performance now has a CPE of 2.5. How do you explain this performance improvement?

Problem 7. (12 points):

3M decides to make Post-Its by printing yellow squares on white pieces of paper. As part of the printing process, they need to set the CMYK (cyan, magenta, yellow, black) value for every point in the square. 3M hires you to determine the efficiency of the following algorithms on a machine with a 2048-byte direct-mapped data cache with 32 byte blocks.

You are given the following definitions:

```
struct point_color {
    int c;
    int m;
    int y;
    int k;
};

struct point_color square[16][16];
register int i, j;
```

Assume:

- `sizeof(int) = 4`
- `square` begins at memory address 0
- The cache is initially empty.
- The only memory accesses are to the entries of the array `square`. Variables `i` and `j` are stored in registers.

A. What percentage of the writes in the following code will miss in the cache?

```
for (i=0; i<16; i++){
    for (j=0; j<16; j++) {
        square[i][j].c = 0;
        square[i][j].m = 0;
        square[i][j].y = 1;
        square[i][j].k = 0;
    }
}
```

Miss rate for writes to `square`: ________ %

B. What percentage of the writes in the following code will miss in the cache?

```
for (i=0; i<16; i++){
    for (j=0; j<16; j++) {
        square[j][i].c = 0;
        square[j][i].m = 0;
        square[j][i].y = 1;
        square[j][i].k = 0;
    }
}
```

Miss rate for writes to `square`: _________ %

C. What percentage of the writes in the following code will miss in the cache?

```
for (i=0; i<16; i++){
    for (j=0; j<16; j++) {
        square[i][j].y = 1;
    }
}
for (i=0; i<16; i++) {
    for (j=0; j<16; j++) {
        square[i][j].c = 0;
        square[i][j].m = 0;
        square[i][j].k = 0;
    }
}
```

Miss rate for writes to `square`: _________ %