Namn:_____

Person-nummer:_____

Datorarkitektur, 2005

Tentamen 2005-04-02

Instructions:

- Make sure that your exam is not missing any sheets, then write your name and person-nummer on the front. If you need extra pages be sure to write on those too.
- 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.
- 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. (10 points):

In the following questions assume the variable `a` is a signed integer and that the machine uses two's complement representation. Also assume that `MAX_INT` is the maximum integer, `MIN_INT` is the minimum integer, and `W` is one less than the word length (e.g., `W` = 31 for 32-bit integers).

Match each of the descriptions on the left with a line of code on the right (write in the letter). You will be given 2 points for each correct match.

1. `a%2`.

   _____

2. Two's complement of `a`.

   _____

3. `a^a`.

   _____

4. `!a`.

   _____

5. `(a > 0) ? 1 : 0`.

   _____

a. `~((a | (~a + 1)) >> W) & 1`

b. `((a < 0) ? a + 1 : a) >> 1`

c. `~a - (MIN_INT + MAX_INT)`

d. `!((a >> W) | !a)`

e. `(~((~0) << 1)) & a`

f. `(MAX_INT & a) + (~0)`

g. `~((a ^ (~0)) | a)`

Problem 2. (12 points):

Consider the following 16-bit floating point representation based on the IEEE floating point format:

- There is a sign bit in the most significant bit.
- The next seven bits are the exponent. The exponent bias is 63.
- The last eight bits are the significand.

The rules are like those in the IEEE standard (normalized, denormalized, representation of 0, infinity, and NAN).
We consider the floating point format to encode numbers in a form:

$$(-1)^s \times m \times 2^E$$

where m is the *mantissa* and E is the exponent.
Fill in the table below for the following numbers, with the following instructions for each column:

Hex: The 4 hexadecimal digits describing the encoded form.

m**:** The fractional value of the mantissa. This should be a number of the form x or x/y, where x is an integer, and y is an integral power of 2. Examples include: 0, 67/64, and 1/256.

E**:** The integer value of the exponent.

Value: The numeric value represented. Use the notation x or $x \times 2^z$, where x and z are integers.

As an example, to represent the number $7/2$, we would have $s = 0$, $m = 7/4$, and $E = 1$. Our number would therefore have an exponent field of `0x40` (decimal value $63 + 1 = 64$) and a significand field `0xC0` (binary 11000000_2), giving a hex representation `40C0`.
You need not fill in entries marked "—".

Description	Hex	m	E	Value
-0				-0
Smallest value > 1				
Largest Denormalized				
$-\infty$		—	—	$-\infty$
Number with hex representation `3AA0`	`3AA0`			

Problem 3. (8 points):

Consider a **5-bit** two's complement representation. Fill in the empty boxes in the following table. Addition and subtraction should be performed based on the rules for 5-bit, two's complement arithmetic

Number	Decimal Representation	Binary Representation
Zero	0	
n/a	-2	
n/a	9	
n/a	-14	
n/a		`0 1100`
n/a		`1 0100`
TMax		
TMin		
TMin+TMin		
TMin+1		
TMax+1		
−TMax		
−TMin		

Problem 4. (8 points):

Consider the source code below, where `M` and `N` are constants declared with `#define`.

```
int mat1[M][N];
int mat2[N][M];

int sum_element(int i, int j)
{
  return mat1[i][j] +  mat2[i][j];
}
```

A. Suppose the above code generates the following assembly code:

```
sum_element:
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%eax
        movl 12(%ebp),%ecx
        sall $2,%ecx
        leal 0(,%eax,8),%edx
        subl %eax,%edx
        leal (%eax,%eax,4),%eax
        movl mat2(%ecx,%eax,4),%eax
        addl mat1(%ecx,%edx,4),%eax
        movl %ebp,%esp
        popl %ebp
        ret
```

What are the values of `M` and `N`?

`M` =

`N` =

Problem 5. (4 points):

Consider the following C functions and assembly code:

```
int fun4(int *ap, int *bp)
{
    int a = *ap;
    int b = *bp;
    return a+b;
}

int fun5(int *ap, int *bp)
{
    int b = *bp;
    *bp += *ap;
    return b;
}

int fun6(int *ap, int *bp)
{
    int a = *ap;
    *bp += *ap;
    return a;
}
```

```
pushl %ebp
movl %esp,%ebp
movl 8(%ebp),%edx
movl 12(%ebp),%eax
movl %ebp,%esp
movl (%edx),%edx
addl %edx,(%eax)
movl %edx,%eax
popl %ebp
ret
```

Which of the functions compiled into the assembly code shown?

Problem 6. (8 points):

Condider the following assembly code for a C `for` loop:

```
loop:
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%ecx
        movl 12(%ebp),%edx
        xorl %eax,%eax
        cmpl %edx,%ecx
        jle .L4
.L6:
        decl %ecx
        incl %edx
        incl %eax
        cmpl %edx,%ecx
        jg .L6
.L4:
        incl %eax
        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 the symbolic variables `x`, `y`, and `result` in your expressions below — *do not use register names.*)

```
int loop(int x, int y)
{
   int result;

   for (_____________; ____________; result++ ) {

      __________;

      __________;
   }

   __________;

   return result;
}
```

Problem 7. (10 points):

Consider the following incomplete definition of a C struct along with the incomplete code for a function `func` given below.

```
typedef struct node {

  _________________ x;

  _________________ y;

  struct node *next;

  struct node *prev;

} node_t;
```

```
node_t n;

void func() {

  node_t *m;

  m = ____________________________;

  m->y /= 16;

  return;
}
```

When this C code was compiled on an IA-32 machine running Linux, the following assembly code was generated for function `func`.

```
func:
  pushl %ebp
  movl n+12,%eax
  movl 16(%eax),%eax
  movl %esp,%ebp
  movl %ebp,%esp
  shrw $4,8(%eax)
  popl %ebp
  ret
```

Given these code fragments, fill in the blanks in the C code given above. Note that there is a unique answer.

The types must be chosen from the following table, assuming the sizes and alignment given.

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