CS 4400: Computer Systems

Midterm Exam 1 SAMPLE SOLUTIONS
Fall 2010

1.

Expression	Decimal Representation	7-bit Binary Representation
—	51	[011 0011]
—	−12	[111 0100]
—	[42]	010 1010
—	[-43]	101 0101
`-4 << 5`	[0]	[000 0000]
`38 >> 3`	[4]	[000 0100]
TMin	[-64]	[100 0000]
TMax	[63]	[011 1111]
TMin +1	[-63]	[100 0001]
−TMax	[-63]	[100 0001]
TMax − TMin	[-1]	[111 1111]
TMin − TMax	[1]	[000 0001]

2.

Description	Hex	M	E	Value
Negative zero	[0x20]	—	—	—
Positive infinity	[0x18]	—	—	—
[NaN]	0x3E	—	—	—
—	0x15	[$\frac{13}{8}$]	[1]	[3.25]
—	[0x22]	[$\frac{1}{4}$]	[0]	-0.25
One	[0x08]	[1]	[0]	1.0
Smallest denormalized > 0	[0x01]	[$\frac{1}{8}$]	[0]	[0.125]
Largest normalized > 0	[0x17]	[$\frac{15}{8}$]	[1]	[3.75]

3.
```
asm1: # returns the larger of a, b
pushl %ebp
movl %esp, %ebp
movl 8(%ebp), %edx   # edx: a
movl 12(%ebp), %eax  # eax: b
cmpl %edx, %eax  # if(b >= a), return b
jge .L6
movl %edx, %eax  # else, return a
.L6:
popl %ebp
ret
```

```
asm2:
pushl %ebp
movl %esp, %ebp
movl 8(%ebp), %edx
movl 12(%ebp), %eax
cmpl %eax, %edx
jb .L9
movl %edx, %eax
.L9:
popl %ebp
ret

asm3: # returns smaller of a, b
pushl %ebp
movl %esp, %ebp
movl 8(%ebp), %edx  # edx: a
movl 12(%ebp), %eax  # eax: b
cmpl %edx, %eax  # if(b <= a), return b
jle .L2
movl %edx, %eax  # else, return a
.L2:
popl %ebp
ret
```

C function `baz1` corresponds to assembly-code routine *asm3*

C function `baz2` corresponds to assembly-code routine *asm3*

C function `baz3` corresponds to assembly-code routine *asm2*

4.
```
bar:
        pushl   %ebp
        movl    $1, %eax  # eax:1 (result)
        movl    %esp, %ebp
        movl    12(%ebp), %ecx  # ecx: y
        pushl   %esi
        movl    8(%ebp), %esi  # esi: x
        testl   %ecx, %ecx  # if(y <= 0), goto done
        jle     L4
        xorl    %edx, %edx  # edx: 0 (i)
L5:
        incl    %edx  # i++
        imull   %esi, %eax  # eax: result * x
        cmpl    %edx, %ecx  # if(y != i), goto loop
        jne     L5
L4:
        popl    %esi
        leave
        ret
```

```
for(i = 0, result = 1; i< y; i++)
  result *= x;
```

5. scale:

```
pushl %ebp  # save %ebp
movl %esp, %ebp  # set new frame pointer
subl $8, %esp  # allocate 8 bytes of stack
movl 8(%ebp), %ecx  # ecx: i
movl %ebx, (%esp)  # save %ebx
movl 12(%ebp), %eax  # eax: j
movl 16(%ebp), %edx  # edx: s
movl %esi, 4(%esp)  # save %esi
leal (%eax,%ecx,14), %ebx  # ebx: 14i+j
movl arr1(,%ebx,4), %esi  # esi: arr1 + 4(14i+j)
leal (%eax,%eax,2), %eax  # eax: 3j
leal (%ecx,%eax,4), %eax  # eax: 12j+i
imull %edx, %esi  # esi: s*M[arr1 + 4(14i+j)]
imull arr2(,%eax,4), %edx   # edx: s*M[arr2 + 4(12j+i)]
movl %esi, arr1(,%ebx,4)  # store back to M[arr1 + 4(14i+j)]
movl (%esp), %ebx  # restore %ebx
movl %edx, arr2(,%eax,4)  # store back to M[arr2 + 4(12j+i)]
movl 4(%esp), %esi  # restore %esi
movl %ebp, %esp  # deallocate 8 bytes of stack
popl %ebp  # restore %ebp
ret
```

M = 12, N = 14

6. asm1:

```
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %ecx  # ecx: ptr2
movl 8(%ebp), %edx  # edx: ptr1
movl 16(%ebp), %eax  # eax: ptr3
movl (%ecx), %ecx  # ecx: *ptr2
movl (%eax), %eax  # eax: *ptr3
addl %ecx, (%edx)  # *ptr1 += *ptr2
popl %ebp
ret  # return *ptr3
```

```
asm2:
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %eax  # eax: ptr2
movl 8(%ebp), %edx  # edx: ptr1
movl (%eax), %eax  # eax: *ptr2
addl (%edx), %eax  # eax: *ptr1 + *ptr2
movl 16(%ebp), %edx  # edx:  ptr3
popl %ebp
addl (%edx), %eax  # eax:  *ptr1 + *ptr2 + *ptr3
ret  # return *ptr1 + *ptr2 + *ptr3
```

Assembly-code routine `asm1` corresponds to C function *foo5*

Assembly-code routine `asm2` corresponds to C function *foo1* or *foo2*

7.
```
proc1:
  pushl %ebp
  movl  %esp,%ebp
  movl  8(%ebp),%eax  # x
  movl  8(%eax),%eax  # x -> c
  movl  %ebp,%esp
  popl  %ebp
  ret

  proc2:
  pushl %ebp
  movl  %esp,%ebp
  movl  8(%ebp),%eax  # x
  movl  12(%eax),%eax  # x -> f[1]
  movl  %ebp,%esp
  popl  %ebp
  ret

proc3:
  pushl %ebp
  movl  %esp,%ebp
  movl  8(%ebp),%eax  # x
  movl  4(%eax),%eax  # x -> b.i
  movl  20(%eax),%eax  # x -> b.i -> f[3]
  movl  %ebp,%esp
  popl  %ebp
  ret

proc4:
  pushl  %ebp
  movl   %esp,%ebp
  movl   8(%ebp),%eax  # x
  movl   (%eax),%eax  # x -> i
  movl   24(%eax),%eax  # x -> i -> g
  movl   (%eax),%eax  # x -> i -> g -> d
  movsbl 1(%eax),%eax  # x -> i -> g -> d -> a[1]
  movl   %ebp,%esp
  popl   %ebp
  ret
```

A: `return x -> c;`
B: `return x -> f[1];`
C: `return x -> b.i -> f[3];`
D: `return x- > i -> g -> d -> a[1];`