

CSci223
Spring 2005
Homework #3
Due: March 28th

1) Name the eight 32-bit registers on a typical Intel processor. (3 points)

2) Describe the purpose of the `%esp` and `%ebp` registers. (4 points)

3) Using the notation described on page 137 of your textbook, show what each of the following instructions will do. (4 points)

Example: `movl 8(%esp), %eax` $R[\%eax] = M[8 + R[\%esp]]$

a) `movl (%edx), %eax`

b) `movl (%edx, %eax), %ebx`

c) `movl 12(%edx, %ebx, 4), %eax`

d) `movl %eax, 16(%edx)`

4) Use one `leal` instruction to perform the requested operations. The value of x is in `%edx` and the value of y is in `%ebx`. Place the result in `%eax`. (4 points)

Example: $5x+2$ `leal 2(%edx,%edx,4), %eax`

a) $9x+7$

b) $4x+y+3$

c) $3y + 5$

5) Describe the operation of the “one-operand” `imull` instruction. (3 points)

6) Describe the four main “condition flags” and tell when they are set. (4 points)

7) What is the value of `%eax` after each set of instructions? For each, the value of `%ecx` is 8, and the value of `%edx` is 3. (4 points)

a) `cmpl %edx, %ecx`
`setle %al`
`movzbl %al, %eax`

b) `cmpl %edx, %ecx`
`setg %al`
`movzbl %al, %eax`

c) `cmpl %ecx, %edx`
`setne %al`
`movzbl %al, %eax`

8) Convert the following C code into assembly. (5 points)

```
int x = 7;
int r;
if( x < 10 ) {
    r = 1;
} else {
    r = 2;
}
```

9) All for loops can be instead written as while loops. Translate the following for loop into a while loop. (3 points)

```
int x = 0;
for( i = 0; i < 9; i++ ) {
    x = x * i;
}
```

10) Convert the above `for` loop into assembly. Store `x` in `%eax` and `i` in `%ecx`. (5 points)

11) Many programming languages have an `until` statement. Whereas a `while` statement loops *while* it's conditional expression is true, an `until` statements loops *until* it's conditional is true. Write assembly code for the following high-level code. Store `x` in `%eax`. (5 points)

```
int x = 0;
until( x > 9 ) {
    x++;
}
```

12) Why are `switch` statements more efficient than `if/else` statements when there are lots of branches? (4 points)

13) Write equivalent C code for the following assembly code. (5 points)

```
.LC0:
    .string "Hello!\n"
.globl main
main:
    pushl   %ebp
    movl    %esp, %ebp
    movl    $1, -4(%ebp)
    cmpl   $0, -4(%ebp)
    je     .L2
    pushl   $.LC0
    call   printf
.L2:
    leave
    ret
```

14) Write equivalent C code for the following assembly. (6 points)

```
.globl f
f:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $4, %esp
    movl    8(%ebp), %eax
    incl    %eax
    movl    %eax, -4(%ebp)
    movl    -4(%ebp), %eax
    leave
    ret

.LC0:
    .string "%d\n"
.globl main
main:
    pushl   %ebp
    movl    %esp, %ebp
    pushl   $0
.L3:
    cmpl    $4, -4(%ebp)
    jle     .L5
    jmp     .L4
.L5:
    movl    -4(%ebp), %eax
    pushl   %eax
    pushl   $.LC0
    call    printf
    movl    -4(%ebp), %eax
    movl    %eax, (%esp)
    call    f
    movl    %eax, -4(%ebp)
    jmp     .L3
.L4:
    leave
    ret
```

15) Describe the `call`, `leave`, and `ret` instructions. Tell what instructions each of them is a shortcut for. (5 points)

16) Define “caller save” and “callee save” registers. Which registers are which on a IA32 processor? (4 points)

17) List the four things that happen just after a function is called and the four things that happen just before a function returns to the caller. (5 points)

18) How many bytes are used by each array? (4 points)

a) `int a[10];`

b) `double b[20];`

c) `char c[15];`

d) `char* d[15];`

19) For each, move the indicated member of the array into `%eax` using a single `movl` instruction. Assume the address of the array is held in `%ecx` and the index is in `%edx`. (5 points)

a) `int a[10];`
`int x = a[4];`

b) `char b[12];`
`char x = b[9];`

c) `char* p[13];`
`char* x = p[3];`

20) Assume `y` is stored at `-4(%ebp)` and `x` is stored at `-8(%ebp)`. Write assembly for the following C code. (5 points)

```
int y = 3;
int* x = &y;
*x = 4;
```

21) Write assembly for the following C code. Assume a is stored at -4(%ebp). You can ignore alignment for this problem. (5 points)

```
struct record {
    int i;
    char j;
    int* k;
};

struct record a;
a.i = 3;
a.j = 'd';
a.k = &a.i;
```

22) Again, write assembly for the C code above. However, this time, take into account four byte alignment requirements. (5 points)

23) Explain why memory alignment is important on a IA32 system. (4 points)