Homework #3 Answer Key
Due: 06/30/2003

1) (20 Points) Translate the following C control structure into equivalent 8088/8086 assembler code:

```c
static int a;
static int b;
static int x;

if (a < b) {
    x = 1;
} else {
    x = 2;
}
```

2) (20 Points) Translate the following C control structure into equivalent 8088/8086 assembler code:

```c
static int a;

do {
    a = a+1;
} while (a < 10)
```

3) (20 Points) Translate the following C control structure into equivalent 8088/8086 assembler code:

```c
static int ix;
static int array[10];

for (ix = 0; ix < 10; ix++) {
    array[ix] = ix;
}
```

4) (20 Points) Translate the following C control structure into equivalent 8088/8086 assembler code:

```c
static int k;
static int x;

switch(k) {
    case 0:
        x = 1;
        break;
    case 1:
        x = 12;
        break;
    default:
        x = 0;
}
```
5) (20 Points) Translate the following C function into equivalent 8088/8086 assembler code:
Implementation note: Although a good optimizing compiler could implement this routine using no memory for local variable storage, don’t perform this optimization. Allocate stack frame space and use this storage for local variables.

```c
int strlen(char * pch)
{
    int cnt;

    cnt = 0;
    while (*pch++ != '\0') {
        cnt += 1;
    }

    return cnt;
}
```
Problem 1:

```c
static int a; /* signed values */
static int b;
static int x;

if (a < b) {
    x = 1;
} else {
    x = 2;
}

mov ax,a
cmp ax,b
jae labl20 ; NOTE: signed compare
mov x,1 ; jge would be an unsigned
jmp labl30 ; compare and is different

labl20: mov x,2
labl30:
```

Problem 2:

```c
static int a;

do {
    a = a+1;
} while (a < 10)

labl10: add a,1
cmp a,10
jb labl10
```

Problem 3:

```c
static int ix;
static int array[10];

for (ix = 0; ix < 10; ix++) {
    array[ix] = ix;
}

mov ix,0
labl10: cmp ix,10
jae labl20
mov ax,ix
mov si,ax
add si,si
mov array[si],ax
inc ix
jmp labl10

labl20:
```
Problem 4:

```c
static int k;
static int x;

switch(k) {
    case 0:
        x = 1;
        break;
    case 1:
        x = 12;
        break;
    default:
        x = 0;
}
```

Solution 1, using IF-THEN-ELSE construct

```assembly
mov ax, k
cmp ax, 0
jnz labl20
mov x, 1
jmp labl90
labl20: cmp ax, 1
jnz labl30
mov x, 12
jmp labl90
labl30: mov x, 0
labl90:
```

Solution 2, using jump table

```assembly
Jmptab  dw labl10
dw labl20

mov ax, k
cmp ax, 0
jb labl30
cmp ax, 1
ja labl30
mov si, ax
add si, si
jmp jmptab[si]
labl10: mov x, 1
labl20: jmp labl90
labl20: mov x, 12
labl90: jmp labl90
labl90:
```
Problem 5:

```c
int strlen(char * pch)
{
    int cnt;
    cnt = 0;
    while (*pch++ != '\0') {
        cnt += 1;
    }
    return cnt;
}
```

```assembly
strlen proc near
    push bp              ; set up stack frame
    mov bp,sp            ; reserve space for locals
    sub sp,2             ; reserve space for locals
    mov [bp-2],0         ; initialize count
    mov si,[bp+4]        ; use SI to hole pch
    labl10: lodsb         ; get next char
    or al,al             ; test for end of string
    jz labl20            ; get out if end
    inc [bp-2]           ; increment char counter
    jmp labl10           ; repeat

labl20: mov ax,[bp-2]   ; set up return value
    mov sp,bp            ; tear down stack frame
    pop bp
    ret

strlen endp
```