1. Study Sixth edition Chapter 6, Section 6.5.

2. Translate the following C program to Pep/10 assembly language. It multiplies two integers using a recursive shift-and-add algorithm. `mpr` stands for *multiplier* and `mcand` stands for *multiplicand*.

```
#include <stdio.h>

int times(int mpr, int mcand) {
    if (mpr == 0) {
        return 0;
    } else if (mpr % 2 == 1) {
        return times(mpr / 2, mcand * 2) + mcand;
    } else {
        return times(mpr / 2, mcand * 2);
    }
}

int main() {
    int n, m;
    scanf("%d %d", &n, &m);
    printf("Product: %d\n", times(n, m));
    return 0;
}
```

   The test

```
if (mpr % 2 == 1)
```

   checks if `mpr` is odd, which it is when its least significant bit is 1. You can test that bit by ANDing `mpr` with the mask `0x0001`, which sets all the bits except the rightmost bit to zero, and then comparing the result to zero with `BREQ`.

   Your assembly language program must contain (1) a documentation section with your name, date, and assignment number at the top of the program, and (2) trace tags for all the variables.

   Name your file *xx*`prob0618a.pep` (all lowercase) where *xx* is your assigned two-digit number. For example, if your two-digit number is 99 you would name it `99prob0618a.pep`. Note that the app will automatically append the file extension `.pep`.

   Hand in your file as an attachment in Canvas under Assignment 18a.

3. Translate the following C program to Pep/10 assembly language. It multiplies two integers using an iterative shift-and-add algorithm.

```
#include <stdio.h>

int product, n, m;

void times(int *prod, int mpr, int mcand) {
    *prod = 0;
    while (mpr != 0) {
        if (mpr % 2 == 1) {
           *prod = *prod + mcand;
        }
        mpr /= 2;
        mcand *= 2;
    }
}

int main () {
    scanf("%d %d", &n, &m);
    times(&product, n, m);
    printf("Product: %d\n", product);
    return 0;
}
```

Your assembly language program must contain (1) a documentation section with your name, date, and assignment number at the top of the program, and (2) trace tags for all the variables.

Name your file *xx*`prob0618b.pep` (all lowercase) where *xx* is your assigned two-digit number. For example, if your two-digit number is 99 you would name it `99prob0618b.pep`. Note that the app will automatically append the file extension `.pep`.

Hand in your file as an attachment in Canvas under Assignment 18b.