

In-lab 9 Preparation

To prepare for In-lab Exercise 9, look over the process for designing, coding, linking, and testing a small assembly language program. You are going to copy the files from Exercise 8 and modify the Add function to save and restore some registers, and to use a stack frame for at least three words of local variables. You should also read over the lecture notes on the stack and stack frame, and the instructions `move.w`, `movem.l`, `link`, and `unlk` plus the addressing modes `-(sp)` and `(<disp>, sp)`.

You will modify copies of the two source files, your calling function and your adding function. You will also need a Link Control File (`.lcf`) to link the object output from the two assemblies together with the `bdblib` object file.

Your calling function will resemble this fragment of C++ even more closely, since stack frames and register saving are standard features of higher-level languages:

```
int num1 = 47;
int num2 = 53;
...
cout << " is " << add(num1, num2) << endl;
```

And your adding function (again, closer to reality):

```
int add(int a, int b)
{
    int x = a, y = b, sum;    // 3 local variables

    cout << "sum of " << x << " and " << y;

    sum = x + y;
    return sum;
}
```

You can translate this sample code to PDL and then translate the PDL into M68000 Assembly language. This is a better approach than converting straight from C++ to Assembly. Be sure to draw a map of your stack frame, and to use `.EQU` statements to reference both your stack arguments and your local automatic (stack) variables.