CS 264 Systems **Spring 2022**
http://comp264.org

Stack Lab

Name:

Function calls rely on a very simple data structure called a stack to keep track of who called them. The stack is like a trail of bread crumbs that the program can use to figure out where it should return to. The stack allows you to call the same function from many different places in your program, and it always knows how to get back to where it was called from when it returns. The stack functions like a stack of plates. Every time we call a function, we put a new plate on the stack. The plate has written on it the address of the next instruction after the function call.

In this activity you're going to be drawing stack frames for several functions. Your stack frame should consist of (1) the function's return address, (2) the link register, and (3) space for its variables. In the table below, I have listed the sizes of commonly used datatypes.

Datatype	Size (bytes)
`int`	4
`unsigned int`	4
`uint32_t`	4
`int32_t`	4
`short`	2
`unsigned short`	2
`uint16_t`	2
`int16_t`	2
`char`	1
`unsigned char`	1
`uint8_t`	1
`int8_t`	1

Below I have pasted in the function definition of a few real C functions from one of my lab's research projects.

```
int max_u16(int *buf, unsigned int n) {
    unsigned int k;
    unsigned int maxval = 0;

    ...
}
```

Draw the stack frame for this function below:

Write the prologue of thie function below:

Write the assembly instruction to read the value of `n` into register `D0`:

Write the assembly instruction to write the value in `D2` into register the variable `k` on the stack:

```
void bit_seq_extract_this(int jump, int key_length){
    int i, bin_length,  bin_ptr, bin_num;
    int16_t noise_this[200];

    ...
}
```

Draw the stack frame for this function below:

Write the prologue of thie function below:

Write the assembly instruction to read the value of `bin_length` into register `D0`:

Write the assembly instruction to write the value in `D2` into register the variable `jump` on the stack:

```
void key_renconcile_this(){
    int best_hamming = 0x00;
    int best_hamming_dis = 7;
    unsigned int i;

    ...
}
```

Draw the stack frame for this function below:

Write the prologue of thie function below:

Write the assembly instruction to read the value of `best_hamming_dis` into register `D0`:

Write the assembly instruction to initialize the `best_hamming` to 0 on the stack:

```
void key_reconcile_other(){
    int temp[7];
    unsigned int i;
    int best_hamming = 0x00;
    int best_hamming_dis = 7;
    int temp_bit = 0;

    ...
}
```

Draw the stack frame for this function below:

Write the prologue of thie function below:

Write the assembly instruction to read the value of `temp[0]` into register `D0`:

Write the assembly instruction to initialize the `best_hamming` to 0 on the stack: