322 Compilers: Assignment 5
Code Generation

Your Job

Write a code generator that accepts a series of statements, as produced in assignment 4 and produces a standalone `a.out` that runs on the T-lab machines.

Runtime System

The runtime system (code that the compiled code uses) consists of one function for each of the functions that can be called, i.e., `printint`, `printstr`, `printant`, and `allocate`. Here is the implementation of `printstr` that matches the version from `evalil`.

```
void pc(int i,int pos) {
  printf("%c",(i >> (pos*8)) & 0xFF);
}

void printstr(int *p) {
  int size = *p;
  p++;
  int c=0;
  while (1) {
    if (c==size) break; pc(*p,3); c++;
    if (c==size) break; pc(*p,2); c++;
    if (c==size) break; pc(*p,1); c++;
    if (c==size) break; pc(*p,0); c++;
    p++;
  }
  printf("\n");
}
```

Submission Instructions

Submit a single `zip` file containign your test cases in directory called `5a` and your code in a directory called `5b`. The `5b` directory should contain a script called `tc` that accepts the filename of a tiger program on the commandline and produces a file (in the current directory) called `a.out` that runs the tiger program. The Tiger filenames must follow the same conventions as in assignment 4.

Your `zip` file should also contain subdirectories `1a`, `1b`, `2a`, `2b`, `3a`, `3b`, `4a`, and `4b` containing either your submissions from last time, or fixed versions of them. The revised implementations (but not the revised test cases) will be used when we re-run the test fests.

Tips

- Assign each temporary variable a location on the stack; for each operation, bring the values into a register, do the operation and stick them back into the temporary location on the stack.
- Write a runtime system as a single file `runtime.c` containing one function for each of the functions that can be called, plus a main function that calls a function generated by your compiler.
  ```
  gcc -O2 -c -o runtime.o runtime.c
  ```
- Generate assembly code and use `as` to assemble it. If you generate assembly in the file `prog.S`, issue this command:
  ```
  as -o prog.o prog.S
  ```
- Use `gcc` to link it with your runtime system to build an executable.
  ```
  gcc -o a.out prog.o runtime.o
  ```
- Use `gcc -S` to see example assembly code (specifically to build a skeleton for your generated assembly files).
- Beware: Mac OS X machines have to have the stack always aligned on 16 word boundaries.

Speed contest

Submit one test file called `intspeed.tig` that will be run in all implementations to test the speed of their compiled code. Your compiler must produce a binary that runs in less than 5 seconds for your `intspeed.tig` (on the t-lab machines) in order for your `intspeed.tig` to be considered in the contest.