CS406: Compilers

Homework Assignment 2 - Activation Record, Register Allocation, Instruction Scheduling
Due: 18/3/2022

The objective of this assignment is to practice applying the concepts underlying:

1. Function Call Management
2. Optimizations involving register allocation and instruction scheduling.

1 Problem Set

1. (4 points) Assume that the following program is running on a 64-bit machine with 4 registers (each register is 64-bit wide.). The calling convention uses caller saves. Assume addresses (i.e., pointers), floats, and doubles are 8 bytes, and ints are 4 bytes. Draw the complete stack (i.e., the stack including all active activation records starting from main function) for the program right after *foo has called bar, and before bar returns.* For each slot in the stack, indicate what is stored there, and how much space that slot takes up. Also indicate the activation record that the slot belongs to. Assume that no temporaries are required.

```
1  //initialize the below string with your first name
2  //char *fname = "Nikhil";
3  void main() {
4  int x;
5  double y;
6  float f;
7  //assume x, y initialized.
8  x = foo(y + f);
9  //assume that the following instruction is located at address: 0x10001A1B
10 printf("Done");
11 exit(0);
12 }
13
14 float foo(int a) {
15 g = bar(a,fname);
16 //assume that the following instruction is located at address: 0x10002A2B
17 return g;
18 }
19
20 double bar(int a, char* r) {
21 float h;
22 h = 1.0 * (strlen(r) + s);
23 if (r == s) {
24  double q = 2.0;
25  h = h * q;
26 }
27 return h;
28 }
```

2. (6 points) Perform bottom-up register allocation on the code below for a machine with three registers. Show what code would be generated for each 3AC instruction. When choosing registers to allocate, always allocate the lowest-numbered register available. When choosing registers to spill, choose the non-dirty register that will be used farthest in the future. In case all registers are dirty, choose the register that will be used farthest in the future (note that while determining the distance, you have to consider the order in which the operand is loaded from memory.). In case of a tie, choose the lowest-numbered register.

```
1   A = B + C
2   C = A + B
3   T1 = B + C
4   T2 = T1 + C
5   D = T2
6   E = A + B
7   B = E + D
8   A = C + D
9   T3 = A + B
10  WRITE(T3)
```

3. (6 points) Given the following resource constrains:
   - 2 ALUs (fully pipelined) and one LD/ST unit (not pipelined).
   - Either of the ALUs can execute ADD (1 cycle).
   - Only one of the ALUs can execute MUL (2 cycles).
   - LDs take up an ALU for 1 cycle and LD/ST unit for two cycles.
   - STs take up an ALU for 1 cycle and LD/ST unit for one cycle.

   i) Draw the reservation tables. Considering the reource constraints, ii) Draw the DAG for the code shown below. iii) Schedule using height based list scheduling.

```
1   LD A R1
2   LD B R2
3   LD C R3
4   LD D R4
5   R5 = R1 + R2
6   R6 = R5 * R3
7   R7 = R1 + R6
8   R8 = R6 + R5
9   R9 = R4 + R7
10  R10 = R9 + R8
```

2 What you need to submit?

- A single document named `CS406HA2_response_<yourrollnumber>.pdf`. E.g., a student with roll number 10002000 would submit the response file `CS406HA2_response_10002000.pdf` containing:
  - *handwritten* responses to all questions.

3 How to submit?

1. Click on the link shared with you on the discussion forum. This will create a repository in your GitHub account. Clone the repository into your local development environment.
2. Write your responses clearly on a sheet of paper and convert it to digital form (by taking a photo or scanning). Name the file (digitized) as mentioned previously.
3. Add the file to your GitHub local repository using `git add` command.
4. Save the changes using `git commit` command
5. Upload the changes to GitHub using `git push` command
6. Release your changes by first tagging your commit on the *local environment* using `git tag -a cs406ha2submission -m ‘‘<release description>’’`.

Next, push the tag to *GitHub* with the help of the following command: `git push -tags`
If you want to make changes after you have submitted (repeat the above steps from 2 to 5 and apply modified commands shown below in place of step 6):

```
> git tag -a -f cs406ha2submission -m "<release description>"
> git push -f --tags
```

Make sure that the digital versions of hand-written documents are clearly visible. You will loose points for unclear responses.