U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação | 3º YEAR
EIC0028 | Compilers | 2017-2018 – 2º Semester

Open book exam. Duration: 1h30m

Second Midterm Exam ("Mini-Teste")

Group 1. Low-Level Intermediate Representations (5 pts)

Consider the section of C code, the types of the variables, and the storage information indicated below.

```
sum = 0;
for(int i=0; i<N; i++) {
   sum += A[i];
}
mean = sum / N;
```

1.a) [3pt] Draw an LLIR[1] for the section of code considering the *Jouette+* (instruction set presented in annex) as the target microprocessor.

1.b) [2pt] Explain the possible templates that can be used to generate the LLIR for implementing FOR loops and justify their advantages/disadvantages in the context of the FOR loop in this code example.

Variable	Type	Storage
A	int A[N] (array of 32-bit integers)	Base address of the array stored in the stack, position SP + 4
sum	int sum (32-bit scalar variable)	Variable stored in register r4
mean	int mean (32-bit scalar variable)	Variable stored in register r2
i	int i (32-bit scalar variable)	Variable stored in register r3
N	int N (32-bit scalar variable)	Variable stored in register r1

Group 2. Selection of Instructions and Code Generation (5 pts)

2.a) [2pt] Perform the selection of instructions using the Maximal Munch algorithm for the LLIR of **1a)**, indicating the tilling used.

2.b) [2pt] Write the sequence of instructions generated considering the selection of instructions obtained in **2a)**.

2.c) [1pt] Considering the cost for each instruction presented in the table below, indicate the cost resultant of the instruction selection in **2a)**. Indicate the tilling resultant with the use of dynamic programming and the cost obtained.

Instruction	Cost	Instruction	Cost	Instruction	Cost	Instruction	Cost
MADD	5	LOAD	3	ADD	2	MOVEM	4
MUL	3	STORE	2	ADDI	1	Other instructions	1

[1] LLIR: Low-Level Intermediate Representation.

Group 3. Register Allocation (10 pts)

Consider that you are part of a team working in a new compiler and at the moment the team is focused on the register allocator. The team is discussing the criterion to select the variables for spilling. The team is considering the use of equation (1) for each scalar variable:

$$(\#loads + \#stores) / (\#defs + \#uses) \quad (1)$$

3.a) [1pt] Describe the possible intention of the use of the term (#defs + #uses) in equation (1)[2];

3.b) [1pt] Show an example where the use of (#defs + #uses) in the equation makes a different selection than the one using the equation (#loads + #stores) and explain advantages/disadvantages of considering (#defs + #uses);

```
// N and MIN_INT represent integer constants
// A is a constant representing the base address of array A
// M[x] represents a memory load from position x

// Live-in={}
      i=0;
      max = MIN_INT;
      loop1: if(i>=N) goto end1;
      t = A+4*i;
      s = M[t];
      if(s <= max) goto skip;
      max = s;
      skip: i = i+1;
      goto loop1;
      end1:
// Live-out={max}
```

Considering the code on the left, which represents a section of code in a low-level IR, answer the following questions:

3.c) [1pt] Show a control-flow graph (CFG) for the code;

3.d) [2pt] Present the first 2 iterations of "liveness analysis";

3.e) [1pt] Draw by direct inspection of the code, the interference graph for the local scalar variables used in the section of code presented.

3.f) [2pt] Indicate a possible allocation of variables to registers using the graph coloring algorithm explained in class and supposing the utilization of 3 registers (*R1*, *R2* and *R3*). Show the content of the stack immediately after the simplification of the interference graph. In the case of having to consider *spilling*, use equation (1) for the criterion for selection of variables for *spilling* and the order of selection determined by that criterion. In case of *spilling*, show the result of the first graph coloring (i.e., without repeating the process).

3.g) [2pt] Suppose that we would like to include profiling information (e.g., obtained by executing the program with representative input data) to drive the compiler. Describe how do you envision the use of that information to select the variables for possible spilling. How do you think equation (1) can be extended to take into account to that information?

(end.)

2 #load and "stores represent the number of load and store operations executed when running the code. #defs and #uses represent the number of definitions and uses, respectively.