Open book exam. Duration: 45 min. Second Midterm Exam ("Mini-Teste")

Group 1. Low-Level Intermediate Representation (LLIR) and Instruction Selection (7 pts)

1.a) [3pt] Indicate the LLIR for the section of the code in the example Code1 based on the LLIR presented in the lectures of the course, which is based on expression trees, and the type and storage of the variables given by the table below.

1.b) [4pt] Consider a target machine with 32 32-bit registers (R0 stores 0) including the instructions presented in the table below, where Ri, Rk, and Rj represent registers of the machine (from R0 to R31), C represents a 16-bit signed integer constant, and label1 represents a label identifying the target instruction of the jump. Using the Maximal Munch algorithm, present the instruction selection for the LLIR presented in 1.a) when targeting the machine of 1.b) (it is enough to draw in the LLIR the group of nodes for each selected instruction).

Code1:

```
for(i=0; i < 100; i++)
    A[i] = A[i]/sum;
```

Var	Type	Storage
A	int32 A[100] (1D array of 32-bit integers)	Base address of A in the stack at SP + 8
i	int (32-bit scalar variable)	register r3
sum	int (32-bit scalar variable)	register r1

Instruction	Operation
load Ri, Rj, C	Ri ← Mem[Rj+C]
store Rj, C, Ri	Mem[Rj+C] ← Ri
add Ri, Rj, Rk	Ri ← Rj + Rk
addi Ri, Rj, C	Ri ← Rj + C
mul Ri, Rj, Rk	Ri ← Rj * Rk
div Ri, Rj, Rk	Ri ← Rj / Rk
lth Ri, Rj, label1	If(Ri < Rj) goto label1
gte Ri, Rj, label1	If(Ri >= Rj) goto label1
jump label1	goto label1

Code2:

```
    Live-in={a,b}
1.  t1 := a*a
2.  t2 := a*b
3.  t3 := 2;
4.  t4 := t3*t2
5.  t5 := t1+t4
6.  t6 := b*b
7.  t7 := t5+t6
    Live-out={t7}
```

Code3:

```
i1:  mul R1, R8, R8
i2:  mul R2, R8, R9
i3:  addi R3, R0, 2
i4:  mul R4, R3, R2
i5:  add R5, R1, R4
i6:  mul R6, R9, R9
i7:  add R7, R5, R6
```

Group 2. Scheduling and Register Allocation (9 pts)

2.a) [2pt] By only inspecting the code, present the interference graph for Code2, which would result from liveness analysis.

2.b) [3pt] Consider the interference graph (IG) of 2.a). Indicate a possible allocation of registers using the graph coloring based register allocation algorithm presented in the course lectures and considering a maximum of 3 registers (*R1, R2, and R3*). Show the content of the stack immediately after the simplification of the IG. In the case of needing to perform *spilling*, use the degree of each node to decide.

2.c) [4pt] Considering that in the target machine all instructions execute in 1 clock cycle and the machine has one unit for load/store instructions and two units for the other instructions, present the scheduling of instructions resultant of applying the list-scheduling algorithm to the example in Code3. Present the data-dependence graph and the criterion to order the instructions ready for scheduling.

Group 3. Comment the sentences below and justify why you consider each one true or false (4 pts)

3.a) [2pt] It does not matter if you do register allocation before or after scheduling as in any way the code generated will be similar.

3.b) [2pt] When doing dataflow analysis for liveness analysis, we do not consider array variables as it is impossible to determine their liveness analysis using the dataflow analysis considered in the compiler course.

(End.)