TDT4205 Problem Set 6
Spring 2012

PART 1 - Theory due Fri Mar. 13^{th}, 20:00.
PART 2 - Programming due Wed Mar. 18^{th}, 20:00.

Part 1 needs to be passed in order for Part 2 to be evaluated!
ALL answers are to be submitted to *itslearning*

ALL OF THIS ASSIGNMENT IS TO BE DONE INDIVIDUALLY. Cheating ("koking"), including using partial solutions from other students, will cause a failing grade for the whole course. We will be checking for this using a plagerism detecting as well as looking for suspiciously similar submissions.

All submitted source code MUST be able to compile and run on asti. **This assignment counts towards your final grade.** Please read the assignment guidelines on itslearning before starting to work on the assignment. Requests for clarifications can be posted on the itslearning forum.

What to turn in
When turning in assignments, please turn in these files:

- (your_username)_answers.pdf : Answers to non-programming questions (Part 1)
- (your_username)_code.zip,tar.gz,tgz : All your code for this assignment, including makefiles and other necessary code (Part 2)

Part 1 - Theory

Task 1.1 - Optimization (20%)

1. Describe an optimization that is easily and typically done by a modern compiler.

2. Describe an optimization that would be hard for a compiler to do even with dataflow analysis. Explain why.

3. Consider the following program fragment:

```
1  a=1
2  b=2
3  c=3
4  d=a+x
5  e=b+c
6  f=e
7  g=f
8  g=d+y
9  a=b+c
```

   Identify the lines impacted by the optimizations

   (a) Copy propagation

   (b) Common subexpression elimination

   (c) Constant propagation

   and rewrite them appropriately (start with the original program for each case).

Task 1.2 - Misc (20%)

1. Describe in your own words how you can implement bounds checking (a check to see if the index you gave is within the allocated range of the array) when generating code for array lookups / assignments as described in part 2 of this assignment.

2. Pointer arithmetic allows an expression to alter the value of a variable without referring directly to the variable itself. How does this affect the analysis of live variables and available expressions, respectively?

3. How is graph coloring and spilling related to register allocation?

4. Why do compilers often represent basic blocks as DAGs (directed acyclic graphs) in the context of optimization?