ToDo's

1) Write your webpage

- Add: brief description of your research, link to your papers, link to your repositories
- Get yourself a link in http://lac.dcc.ufmg.br/people.html

2) Weekly seminars

- Paper discussions.
- Student presentations.

3) Read a paper, even if partially, once a week.

- Try to summarize the contents of the paper, e.g., two or three sentences.
- Keet the summaries in a RW.txt file. For each paper: [insight/similarities/differences]
- What can I learn with it?
- How does it compare to my work?

4) Read a paper carefully, once a month.

- Choose a paper that is related to your research.
- Prefer papers from good venues
  - Conference: PLDI, POPL, ASPLOS, CGO, CC, ICFP, MICRO, ISCA, OOPSLA, etc
    - If you can, try to find the slides used to present the paper (for conference paper only)
  - Journals: TOPLAS, TACO, IEEE Computers, Communications of the ACM, etc
- Write a careful review. You will use this once the time comes to write your dissertation.
  - Hint: Douglas writes some nice reviews. Take a look into his webpage.

5) Learn English.

- This is by far the most useful skill that any dragon-nester can have.
- Try to read literature in English.
- Try to watch videos in English.
- Try to talk more in the seminars.

Guidelines:

1) Sharpen your teamwork skills.

- Use more the lab.
- Ask your colleagues which kind of project they are pursuing and read their papers.
- Watch PhD and MSc defenses.

- Watch the pre-conference talk of your colleagues; give them suggestions.
- Share ideas, discuss problems, offer help.
- Having lunch with your colleagues is a good way to socialize.

2) Have in mind which conferences and journals are good to you.

- Find related work
- Understand the review process of each venue.
- Understand how to organize papers, and what are context/problem/solution/results
- Learn which conferences are good (http://www.citescholar.org/ & http://shine.icomp.ufam.edu.br/index.php) and where to publish your work.

3) Be attentive to new research ideas

- Lambda the ultimate, blogs, papers
- Talks at the departament. We have a lot of international talks. Try to attend some of them.

4) Study hard.

- Read books. Usually they go much deeper into the details.
- Try to understand deeply the papers more related to your work.
- Whenever possible, try to approximate your research work to the projects you do in classes.

5) Polish your implementation skills.

- Learn about how to set up and run experiments.
- Browse the web for help: wikipedia, stack overflow, etc
- Give your workmates a code review from time to time.

6) Keep your repository neat and clean.

- Use the wiki prodigally. Add lots of explanations.
- Do not deploy modules with commented code.
- Ask your workmates for code reviews.
- If working on LLVM, use its coding standard: www.llvm.org/docs/CodingStandards.html

8) Try to find allies

- Write to authors of papers that you are reading.
- Ask for implementations that other researchers have done before.
- Participate in newsgroups: LLVM, lambda the ultimate, stackoverflow, etc.
- Be always solicitous once someone asks you about a paper that you have published.

9) Know your problem

- Who has been working on it?
- Is it worth publishing?
- Can you explain your contribution in 32 words? What about 16?

10) Know thyself

- Learn about the different research skills.
- Stay Motivated!

Team Work

Science requires a society because even people who are trying
to be good thinkers love their own thoughts and theories
- much of the debugging has to be done by others.

Alan Kay

Today I was talking to Douglas, one of my master students, about a possible project: to determine if a binary is really equivalent to the source code used to compile it. You know: if the compiler is kind of evil, then the binary and the source code might be different. So, we were brainstorming ideas to deal with this problem. The key difficulty was infra-structure. We have the symbolic range analysis, which is quite powerful, and we would like to use it as our weapon of choice to attack the 'equivalence question'. But, our range analysis works on LLVM's IR. I had suggested to poor Douglas to re-implement it on Clang and on the disassembler (like it is so easy to do it...). Then, Henrique, as boldly as it could be, jumped into our discussion, and pointed to us that we could simply convert both - sources and binaries - to LLVM IR. Yes, the thing was as simple as that. Actually, it was shamefully simple. Well, we do not know if this will solve our problem, but it is the idea that we are going to pursue now.

This introduction is just to illustrate the benefits of team work. If we can work together with other people, then we are, in fact, combining many intelligences synergistically. We would not have thought about Henrique's solution immediately, and Henrique was busy with a project of his own, to work on this. But now, we will grab his idea, and will probably extend it with thoughts of our own smarts. This kind of exchange is possible just because we were at the same physical place, trusted each other enough to discuss our research aloud, and have enough liberty with each other to point out something obvious.

Being able to work on a team, of course, has many more benefits than the sharing of ideas. Some of our colleagues might be good partners for the rest of our lives. In the academia, or outside it. This is particularly true in the field of compilers, because there are not many experts on this subject. Sometimes it is almost impossible to find an educated opinion about compilation technology around. Furthermore, we are competing with some very smart people, when we try to publish our stuff. It is hard to fight alone a battle like this. And, since long, it has been very difficult to implement competitive technology, when our rivals are the likes of Google, Intel, Mozilla, and a few other high-tech behemoths. It takes more than one brave warrior to implement anything that beats these guys.

Some people have an easy time grouping with others, but not everybody is like this. Some of us prefer to stay alone. That is ok: each person has a unique mindset. However, I believe that one should always try to improve team skills, for this challenging endeavor might lead to personal growth. There are simple ways to face this difficult task. For a start, ask what

your colleagues are doing. If you do not know anything about their projects, it will be hard to be of any help to them. And see if some of your codes might assist your friends. Sometimes we have already solved a problem that they are facing right now. If someone is using your code, try to be solicitous whenever your aid is asked for. And even if nobody asks you questions about your code, try to make sure that it is easy to read, maintain and use it. You might even ask for a code review. It is often the case that a guest in our house can see trash that we have not observed before. Above all, be proud of your creation, and try to make sure that it will be useful to others. And, in case the situation is the opposite, and you need someone's modules, see if you can contribute something back. Even comments on the sources, or documentation on the wiki, are already useful payments for the help you have received.

If it comes to publishing, try to contribute to the text. Even if you are not one of the authors, see if you can give your friends a review. Who knows if you will not be in need of a review anytime soon? And do not be afraid of confronting the ideas of your colleagues, but if you have to do it, then be polite, and be always ready to accept a reasonable opinion. Nevertheless, do not assume that your friend's idea is the panacea to the problem that you are attacking. Have an opinion too. When two thoughts collide, there is no single victor: each idea is changed a bit during this clash. It is like a confrontation of cultures. For example, we might think that the European civilization completely swallowed the cultures of the native peoples who lived in our Brazil before the Portuguese navigations. Yet, traces of those beautiful and vibrating traditions are still among us. Just look at our foods, language and music. Two ideas exert forces on each other in similar ways. Maybe we will have the impression that one of them has prevailed, but the resulting thought might, quite as well, incorporate vestiges of its early opponent.

Finally, take the opportunity to interact with your colleagues not only as a chance to hone your social skills, but also as an occasion to make friends for life. Our workmates can be good companions, with whom we may discuss not only computer science, but arts, politics, philosophy, everything. We do not have the chance to find intelligent people everywhere, but the university is filled up with them. Besides, politeness, companionship, faithfulness and benignity are values which do not seem to bear too many side-effects.