ACKNOWLEDGEMENT

We express our gratitude to **Dr. B.G. Sangameshwara**. Principal, Sri Jayachamarajendra College of Engineering, for providing an excellent environment for our education and his encouragement throughout our stay in college.

We sincerely thank **Dr. C. N. Ravikumar**, Head of the Department, Computer Science SJCE, Mysore, for providing us with adequate facilities, ways and means by which we were able to complete this project. We express our sincere gratitude to him for his constant support and valuable suggestions without which the successful completion of this project would not have been possible.

We thank **Smt. Vasantha Raghavan**, our guide and **Sri Prashanth Naduthota**, Professor, for their boundless cooperation and helps extended for this project. We express our immense pleasure and thankfulness to all the teachers and staff of Department of Computer Science and Engineering, SJCE for their cooperation and support.

Last but not the least, we thank all others, and especially our classmates and our family members who in one way or another helped us in the successful completion of this work.

Rajkiran Anthapur

Shashidhar G

Vikram T.V

Abstract

The execution of any program is limited by the number of variables that are *live* simultaneously during execution. A variable is said to be *live* across a range during execution if its value will be *used* somewhere later in the program. Registers in the processor hold the live variables and as the number of live variables increases, there will be a shortage of registers to accomodate all the live variables simultaneously. The processor may have to *spill* some of the register values into memory so that the register can be used for some other purpose. Spilling costs execution time because of its access to memory. Instructions can be reordered or scheduled preserving the true data flow order such that the the maximum number of variables that are live simultaneously during program execution gets reduced which is refered to as *register pressure*. Both spilling and scheduling reduces register pressure but there arises a conflict of which of the two should be performed first to have maximum reduction of register pressure.

In our project the MRIS approch is used to reuse the registers by forming *lineages* and reduce the register pressure. The sequencing algorithm then reorders the instructions within the lineages to get the optimal instruction sequnce maintaining the minimal register pressure.