High-level software pipelining in LLVM

Paper Authors:

Roel Jordans, Henk Corporaal

Presented by Group 18:

Yuze Dai, Qinjuan Xie, Yin Yuan, Jiayun Zou

Motivation

Concepts & Course Review

- **Software Pipelining**
  - Loop scheduling
  - Increasing the instruction level parallelism
- **II**: Initiation Interval
- **MaxLive**: Maximum number of simultaneously live values at any cycle
- **Goal**
  - **Higher throughput** (smaller II, smaller stage count)
  - **Lower register requirements** (smaller MaxLive)
- The task of generating an optimal resource-constrained schedule for loops is known to be NP-hard
- Heuristics

Drawbacks of Existing Scheduling Techniques

- Huge Computational Cost
  - Aggressive Schedulings
  - Integer Linear Programming
- Not Considering Critical Path
  - Hypernode Reduction Modulo Scheduling (HRMS)*
- Suboptimal Reduction
  - Stage Scheduling*
- Ejection of Previously Scheduled Operations
  - Slack Scheduling*

*All three schedulings use heuristic technique

Swing Modulo Scheduling (SMS)

Node Ordering

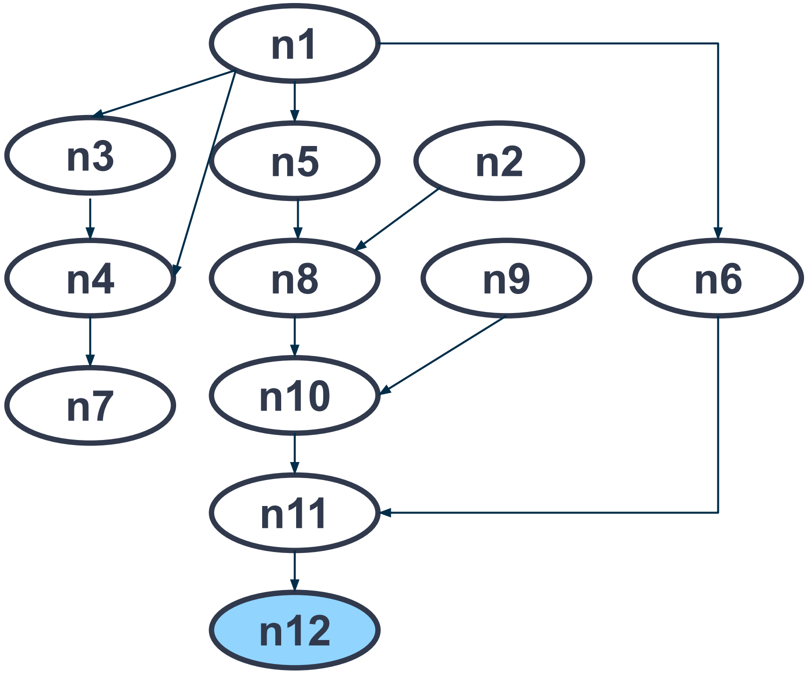
Target

- Give priority to operations in the most critical paths.
- Try to reduce MaxLive

Traversing Algorithm

- Starts by the node at the bottom of the most critical path and moves upwards, visiting all the ancestors
- Once all the ancestors have been visited all the descendants of the already ordered nodes are visited but now moving downwards.
- Successive upwards and downwards sweeps

Example



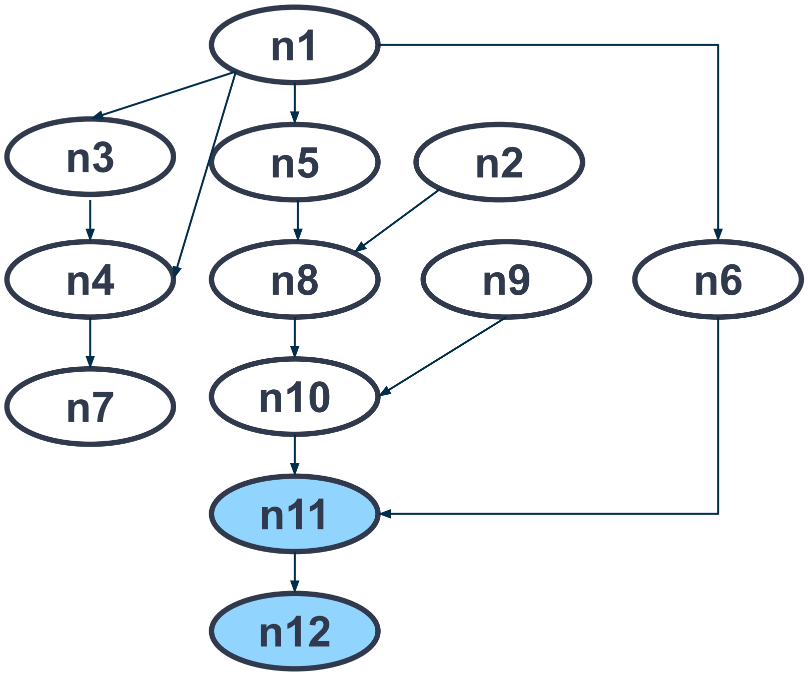
R: set of nodes to be ordered

O: set of nodes been ordered

R={12}

O={12}

Example



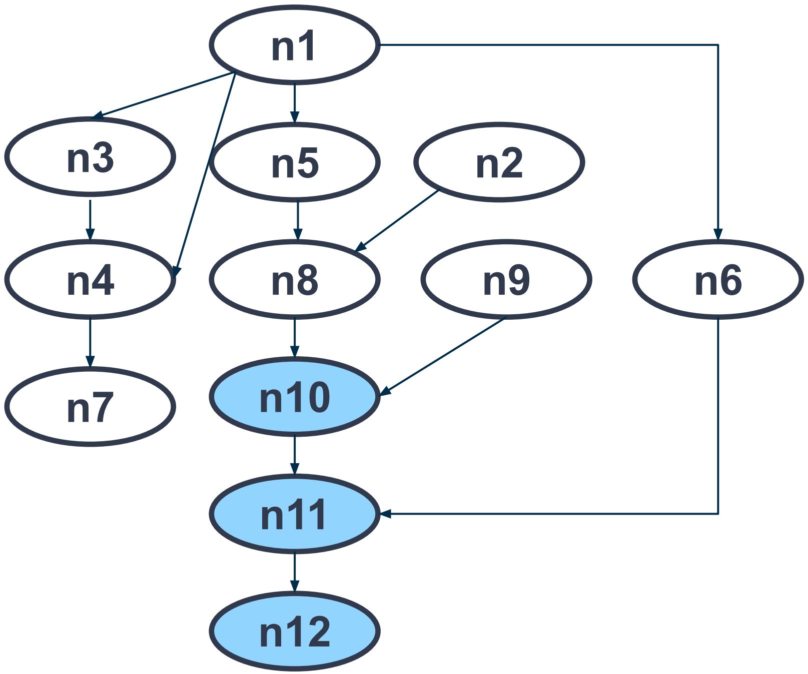
R: set of nodes to be ordered

O: set of nodes been ordered

R={11}

O={12, 11}

Example



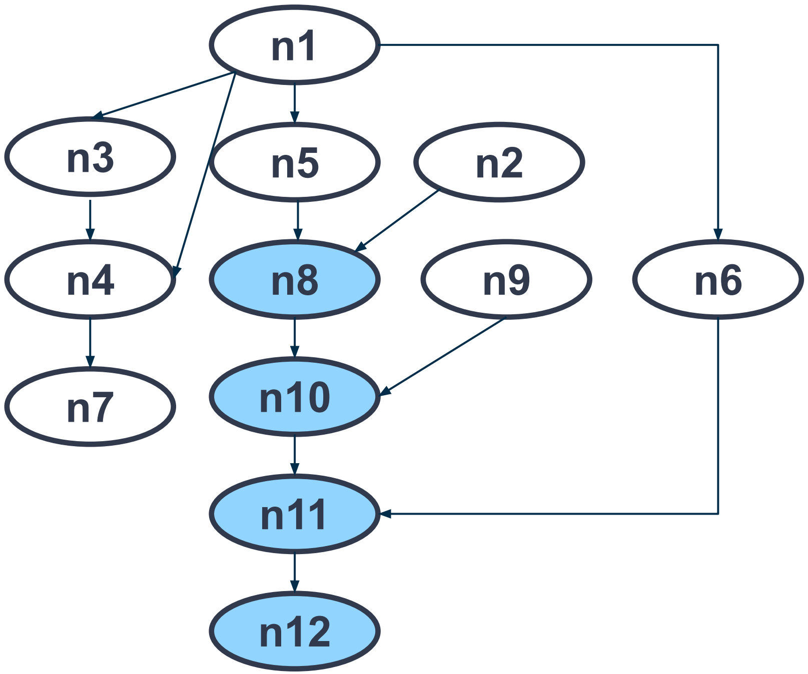
R: set of nodes to be ordered

O: set of nodes been ordered

R={10, 6}

O={12, 11, 10}

Example



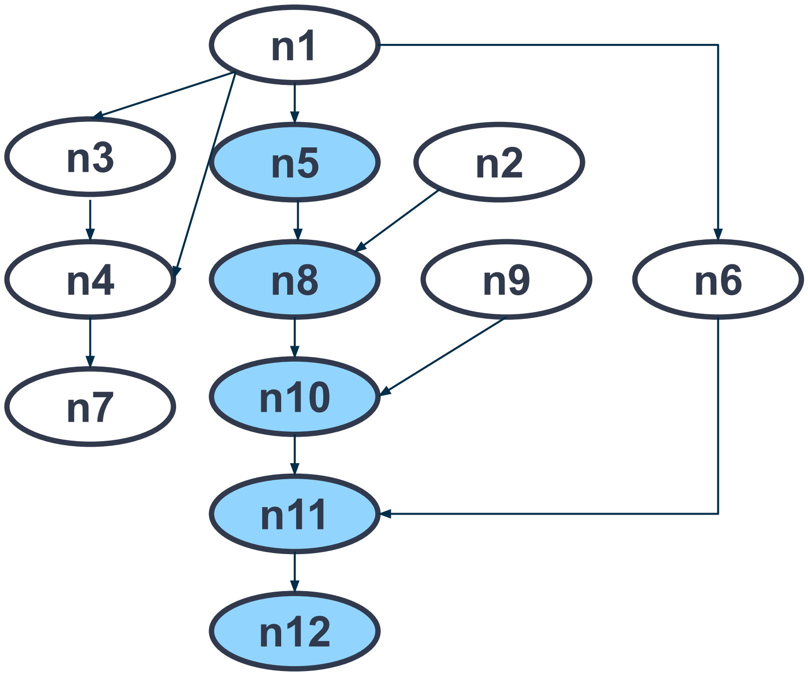
R: set of nodes to be ordered

O: set of nodes been ordered

R={6, 8, 9}

O={12, 11, 10, 8}

Example



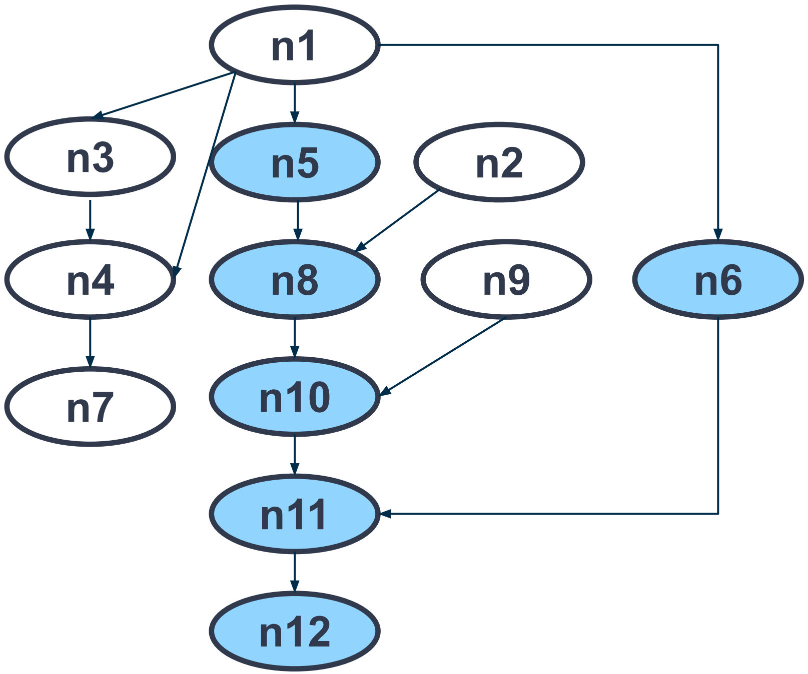
R: set of nodes to be ordered

O: set of nodes been ordered

R={6, 9, 5, 2}

O={12, 11, 10, 8, 5}

Example



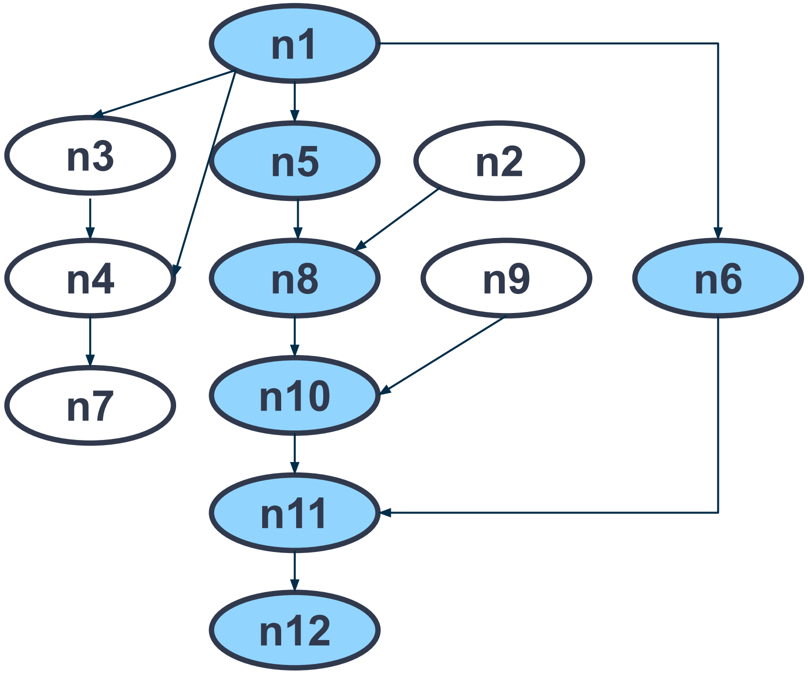
R: set of nodes to be ordered

O: set of nodes been ordered

R={6, 9, 2, 1}

O={12, 11, 10, 8, 5, 6}

Example



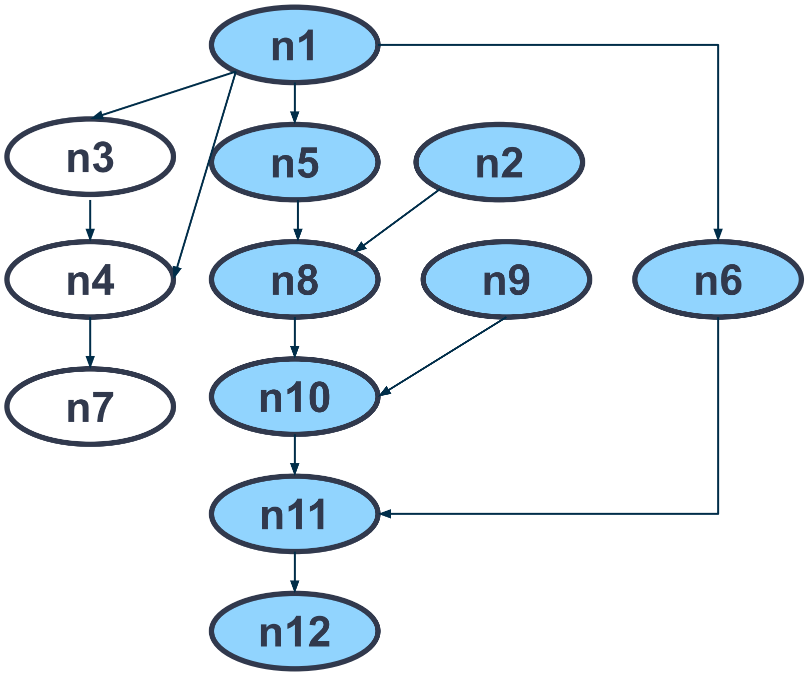
R: set of nodes to be ordered

O: set of nodes been ordered

R={9, 2, 1}

O={12, 11, 10, 8, 5, 6, 1}

Example



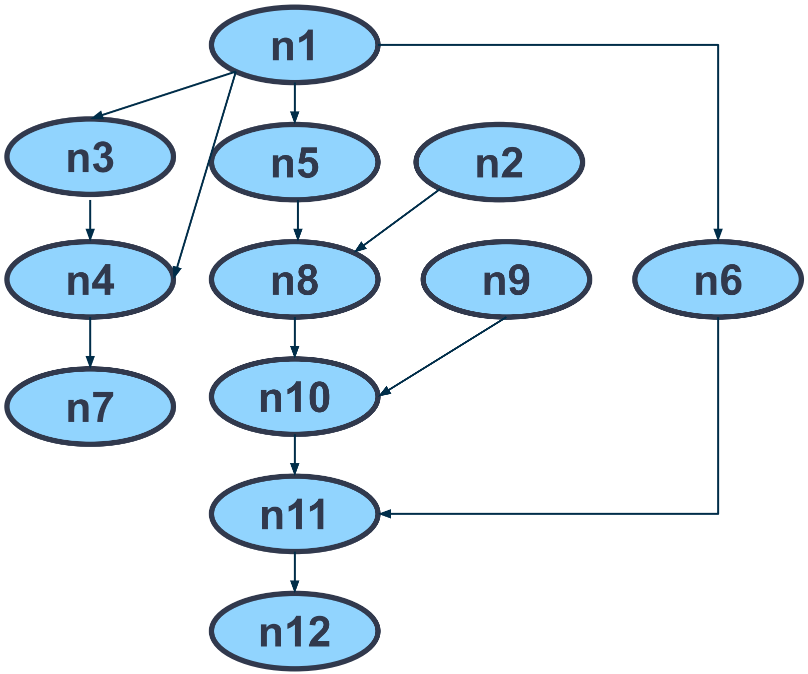
R: set of nodes to be ordered

O: set of nodes been ordered

R={}

O={12, 11, 10, 8, 5, 6, 1, 2, 9}

Example



R: set of nodes to be ordered

O: set of nodes been ordered

R={}

O={12, 11, 10, 8, 5, 6, 1, 2, 9, 3, 4, 7}

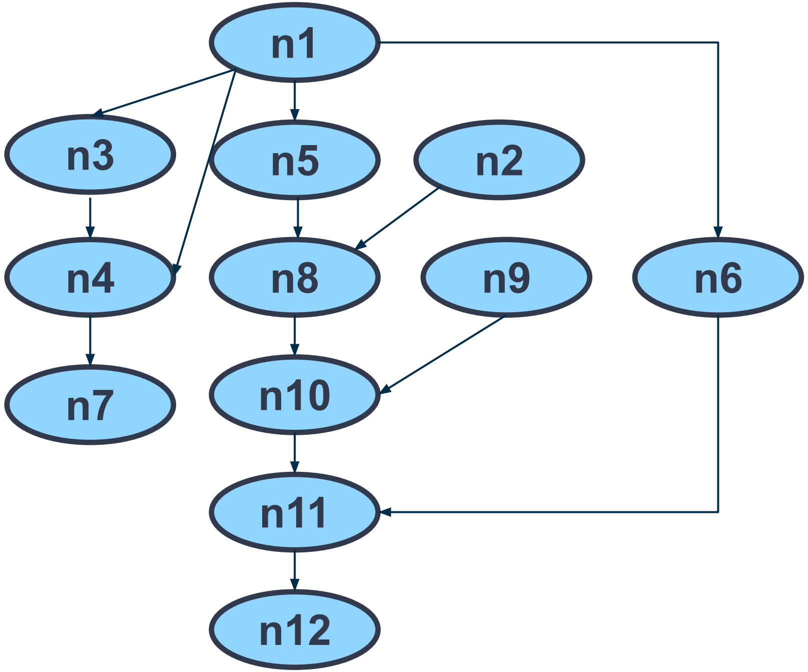
Scheduling

Tries to schedule the operations **as close as possible** to the neighbors that have already been scheduled.

If an operation ***u*** has:

- **Only predecessors** in the partial schedule, then ***u*** is scheduled as soon as possible.
- **Only successors** in the partial schedule, then ***u*** is scheduled as late as possible.
- **Both predecessors and successors**, rare case, only occurs once for each recurrence.

Scheduling, O={12, 11, 10, 8, 5, 6, 1, 2, 9, 3, 4, 7}



	l/s	l/s	add	mul
0	n1			
1				
2	n2		n5	
3				
4			n3	n8
5		n9		
6				n4
7			n10	n6
8		n7		
9				n11
10				
11		n12		

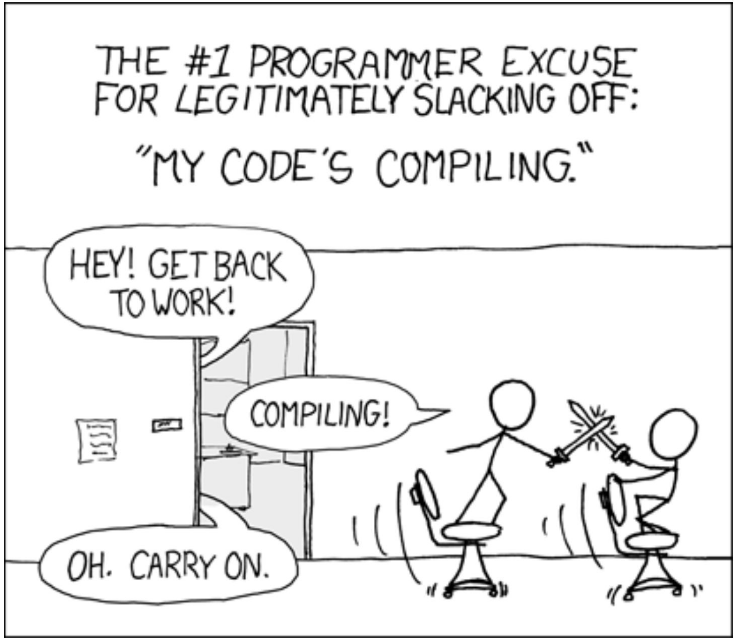
a)

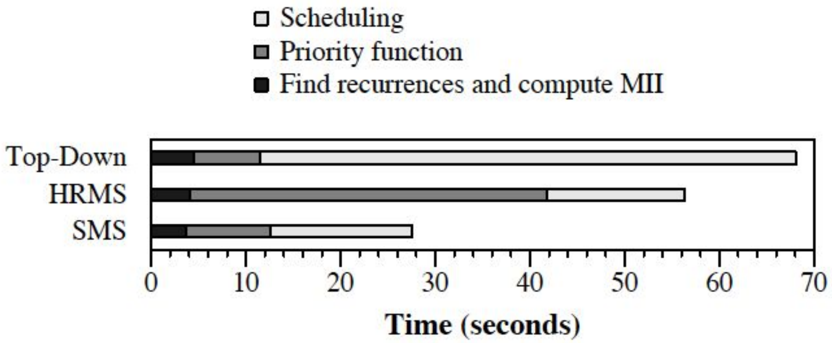
Experiments and Results

Benchmark

- C++ (LEDA libraries)
- Perfect Club benchmark suite without subroutine calls or conditional exits.
- Compared with HRMS(Hypernode reduction modulo scheduling) and Top-Down scheduling.

Compilation Speed

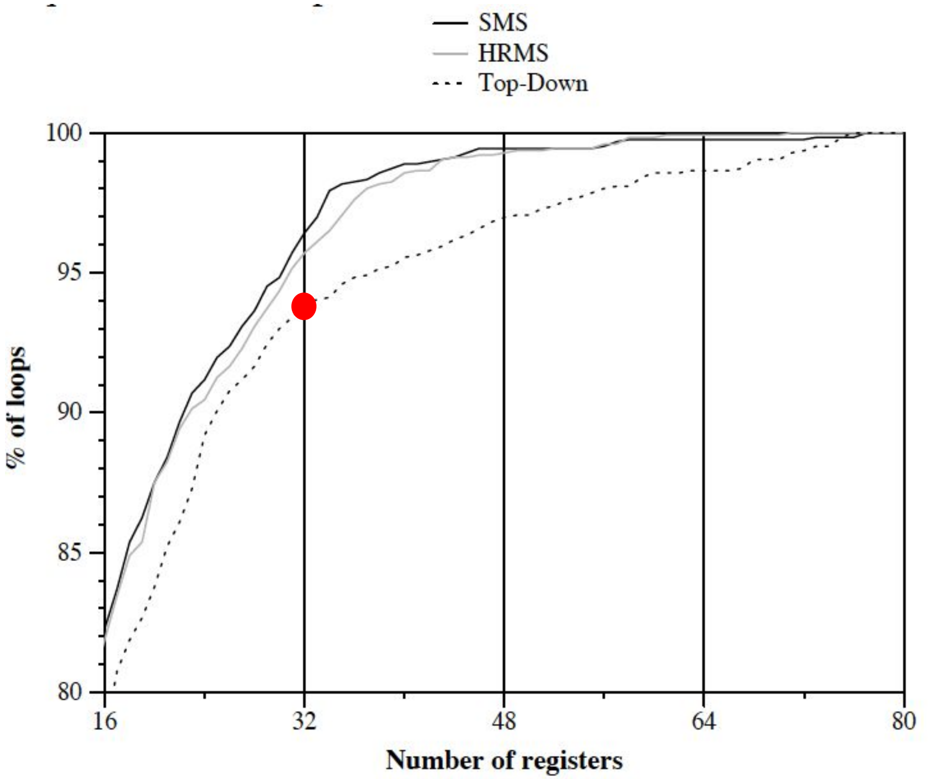




(1258 Loops of the Perfect Club benchmark)

https://xkcd.com/303/

Register Usage



Comparison with Optimal Solution

Program	Loop	Optimal			SMS		
		II	SC	Regs.	II	SC	Regs.
Spice	1	1	3	3	1	3	3
	2	6	3	5	6	3	6
	3	6	1	2	6	1	2
	4	11	2	8	11	2	8
	5	2	2	1	2	2	1
	6	2	12	15	2	12	15
	7	3	7	15	3	7	15
	8	3	2	5	3	2	5
	10	3	2	2	3	2	3
Doduc	1	20	2	5	20	2	7
	3	20	2	3	20	2	4
	7	2	18	18	2	18	18
fppp	1	20	2	2	20	2	2
Livermore	1	3	3	6	3	4	7
	5	3	2	3	3	2	3
	23	9	2	10	9	2	11
Linpack	1	2	3	5	2	3	5
Whetstone	1	17	1	5	17	1	5
	2	6	5	6	6	5	6
	3	5	1	4	5	1	4
	1	4	1	1	4	1	1
	2	4	1	2	4	1	2
	4	4	1	4	4	1	4
	8	4	1	8	4	1	8

Strength and Weakness

Strength

- Produced schedules are very close to the optimal scheduling
- Low computational cost

Weakness

- Required a slight higher registers and stages than optimal schedule
- Missing opportunities for further instruction level parallelism by only handling simple basic block loops

Conclusions

Conclusion

- SMS produces near optimal schedules while requiring a very low compilation time.
- Outperforms other heuristics approaches, which is measured by the attained initiation interval, register requirements and stage count.
- Compares against the optimal solution which was obtained using an integer linear programming approach.
- SMS obtains the initiation interval in all the cases and its schedules requiring only 5% more registers and a 1% higher stage count.

Q&A