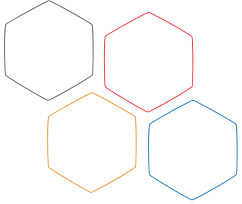
The Little Big Microcontroller (LBMC)

GdR SOC2, Journée thématique du club des partenaires sur le RISCV, Thalès R&T

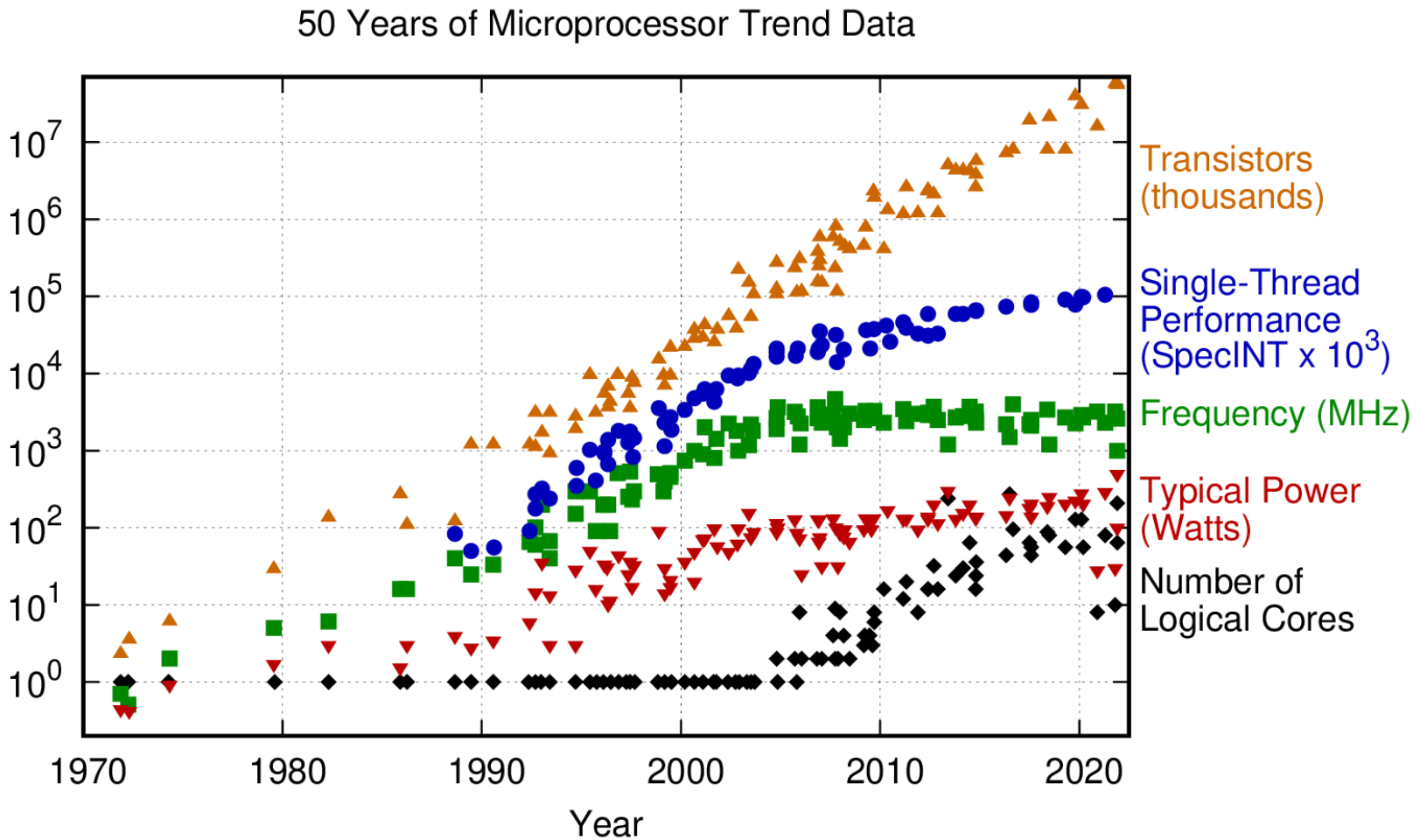
David Parello and Bernard Goossens

29/09/2023



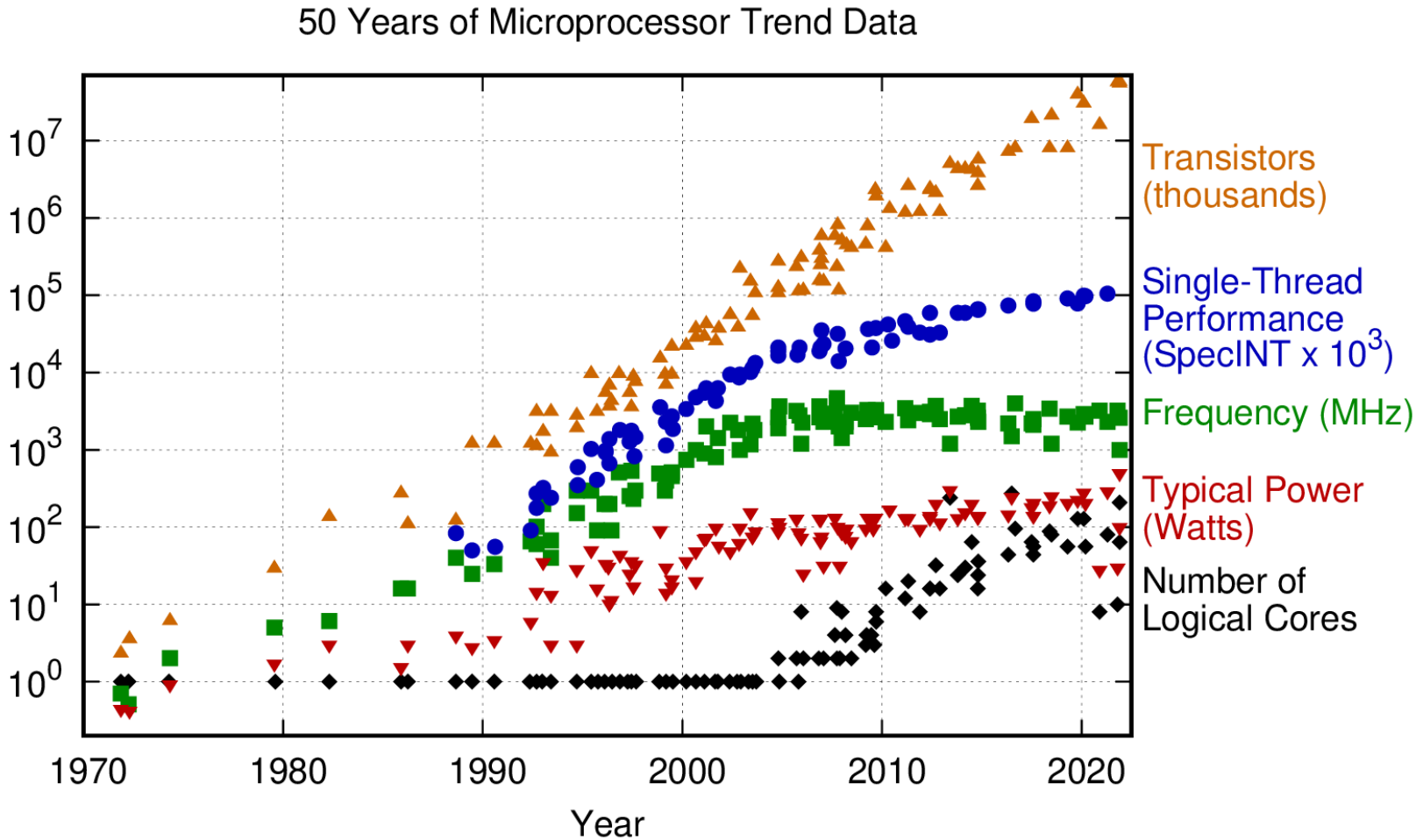
Outline

Microprocessor trends



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

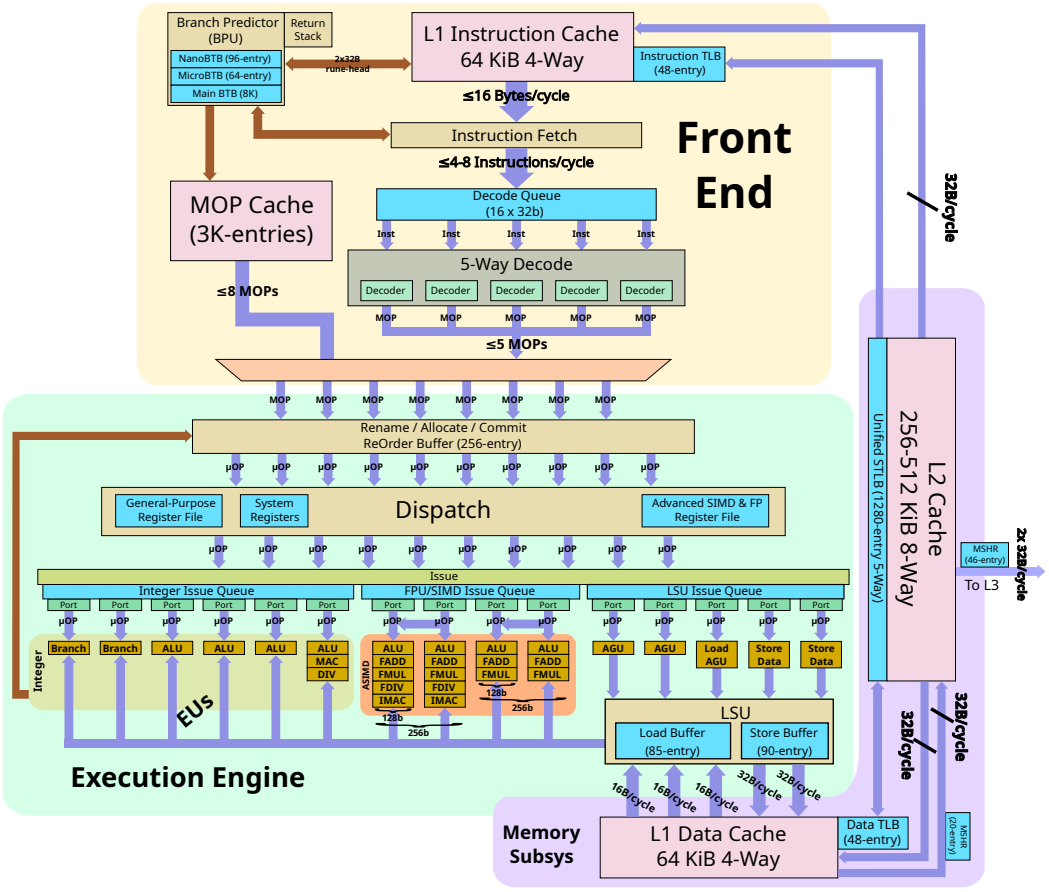
Microprocessor trends



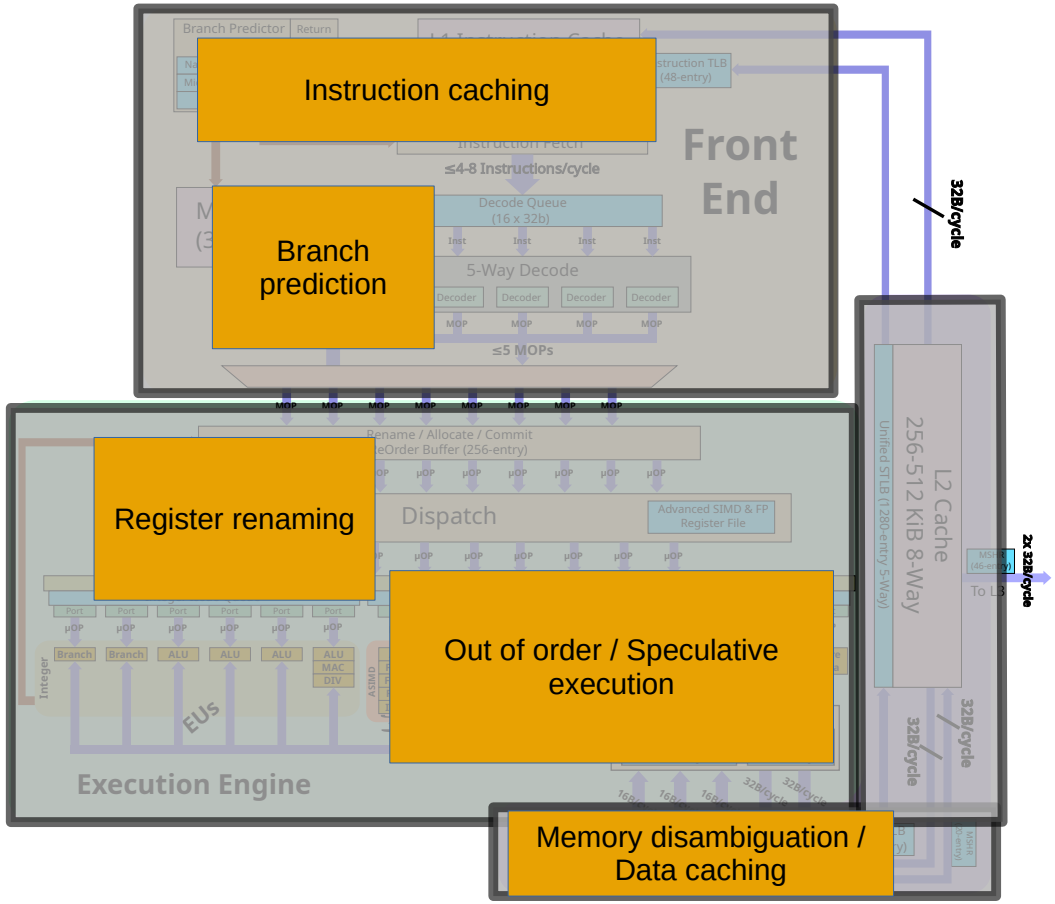
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

PERFORMANCE –> PARALLELISM

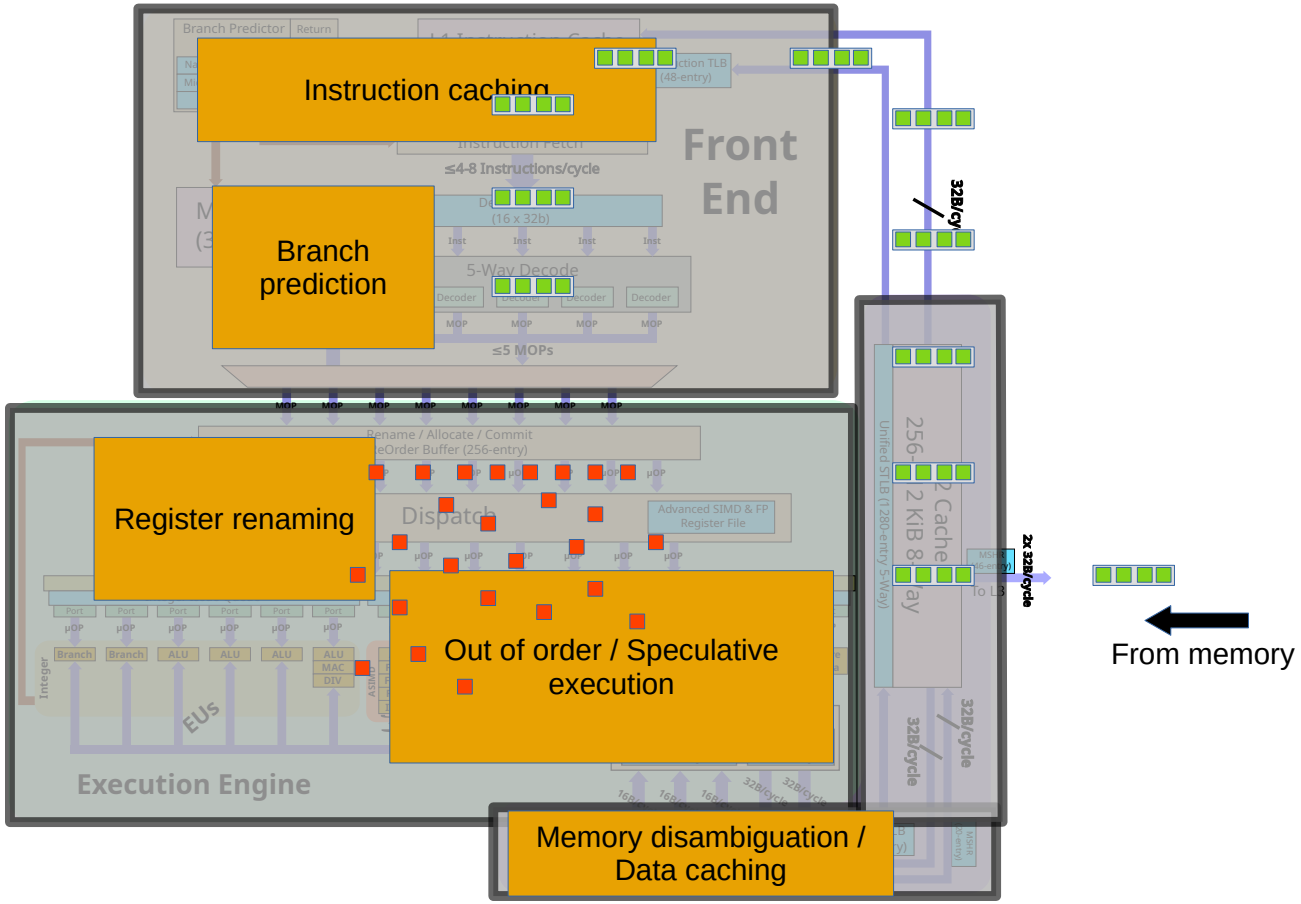
A high performance core (Neoverse v1)



A high performance core

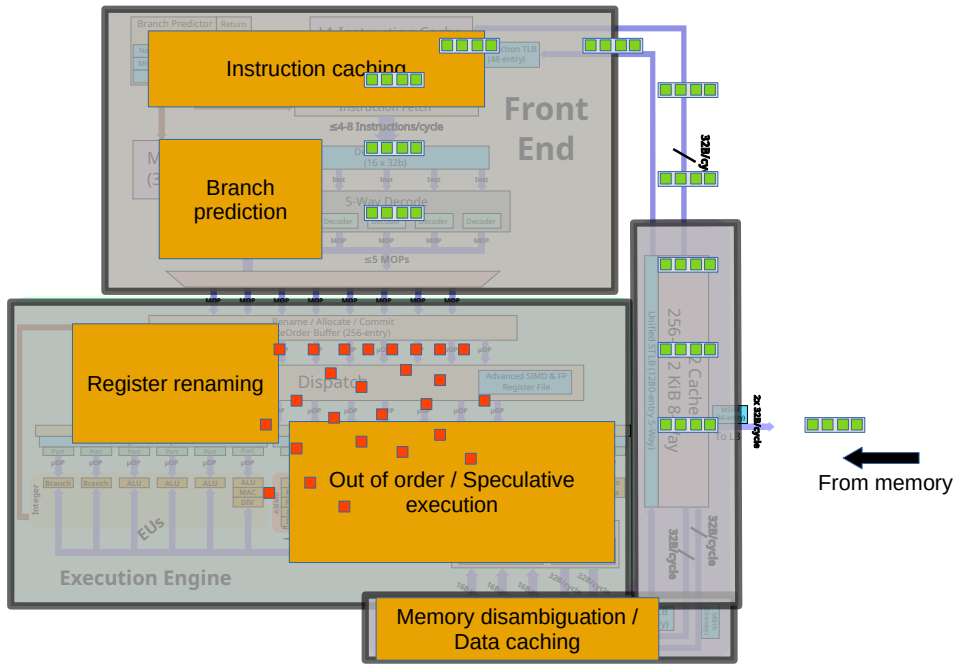


A high performance core



Instruction Level Parallelism (ILP)

HPC Core objective: maximize Single thread performance

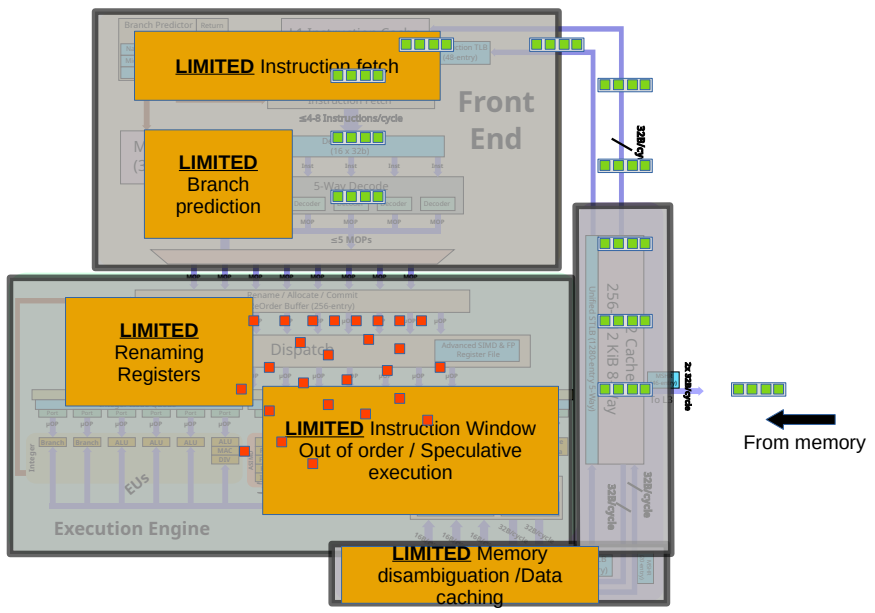


Maxi S.Thread performance => Maxi ILP

- ILP = #Instructions/#Cycles

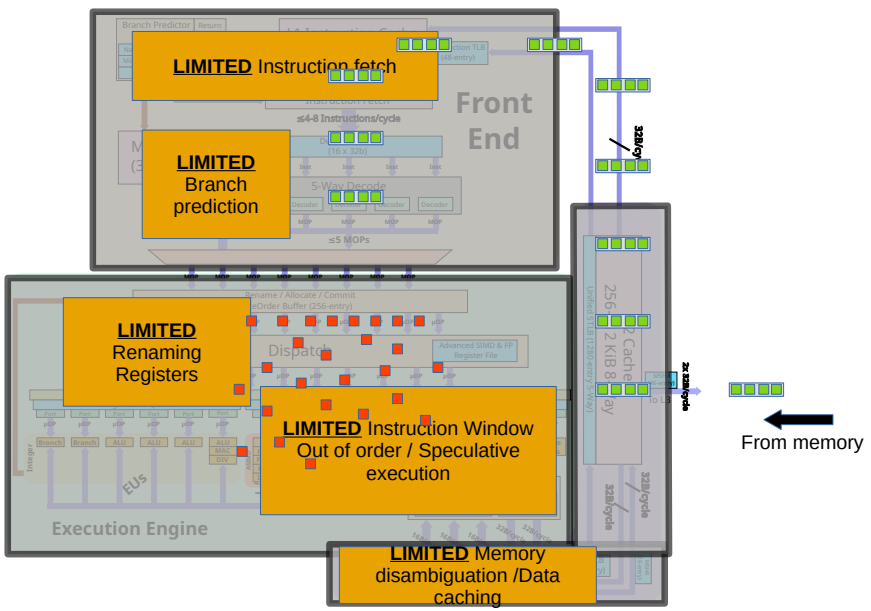
PerPI: an ILP Analyser (Core simulator)

Limited core

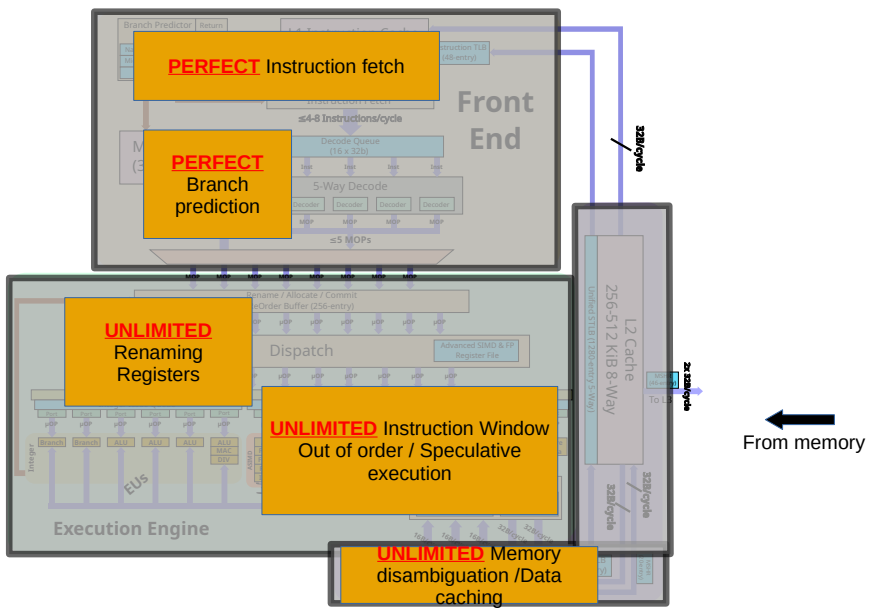


PerPI: an ILP Analyser (Core simulator)

Limited core

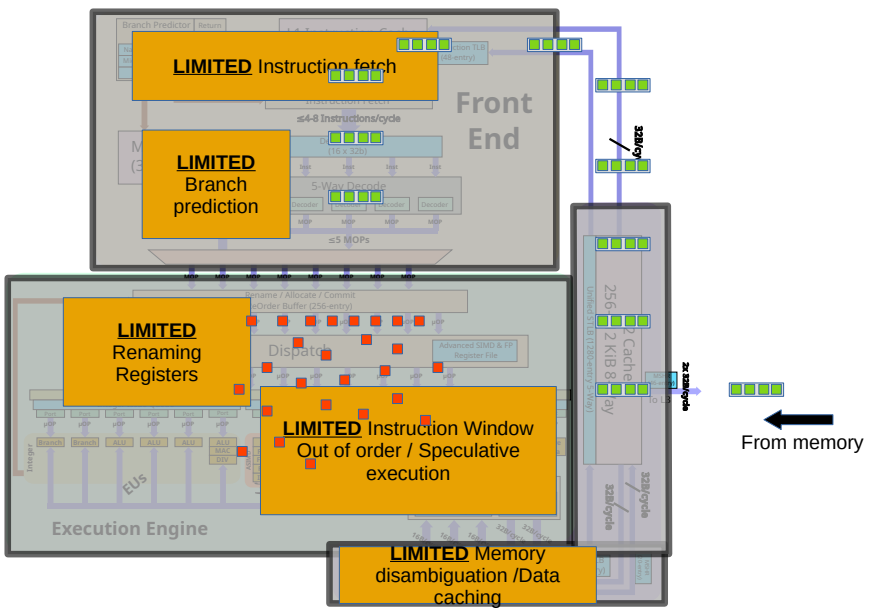


Unlimited core (Ideal)

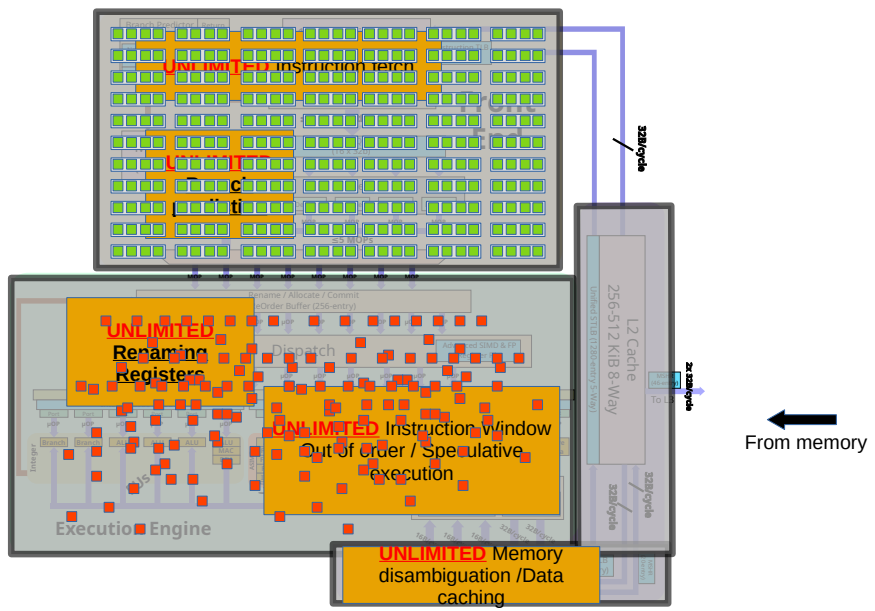


PerPI: an ILP Analyser (Core simulator)

Limited core

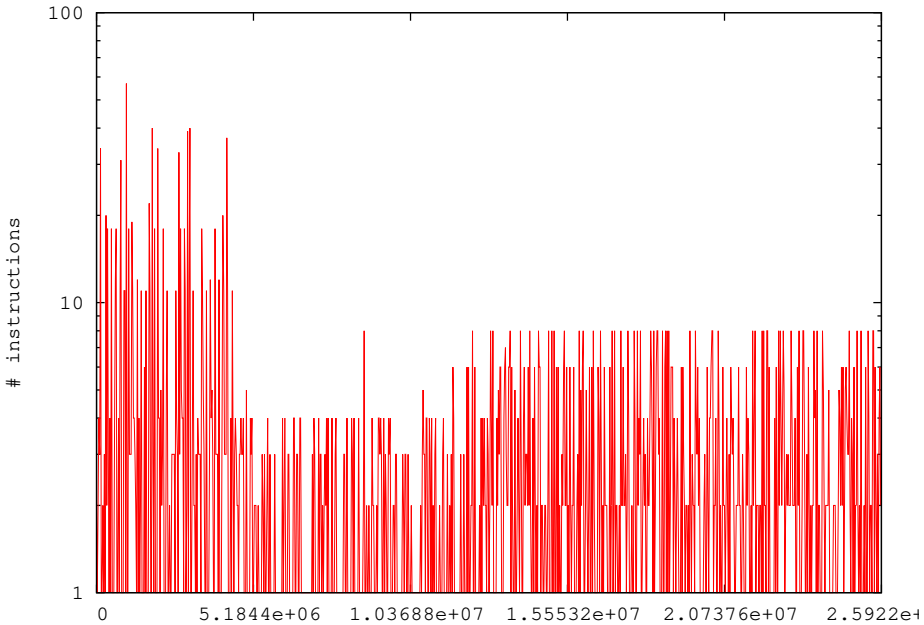


Unlimited core (Ideal)



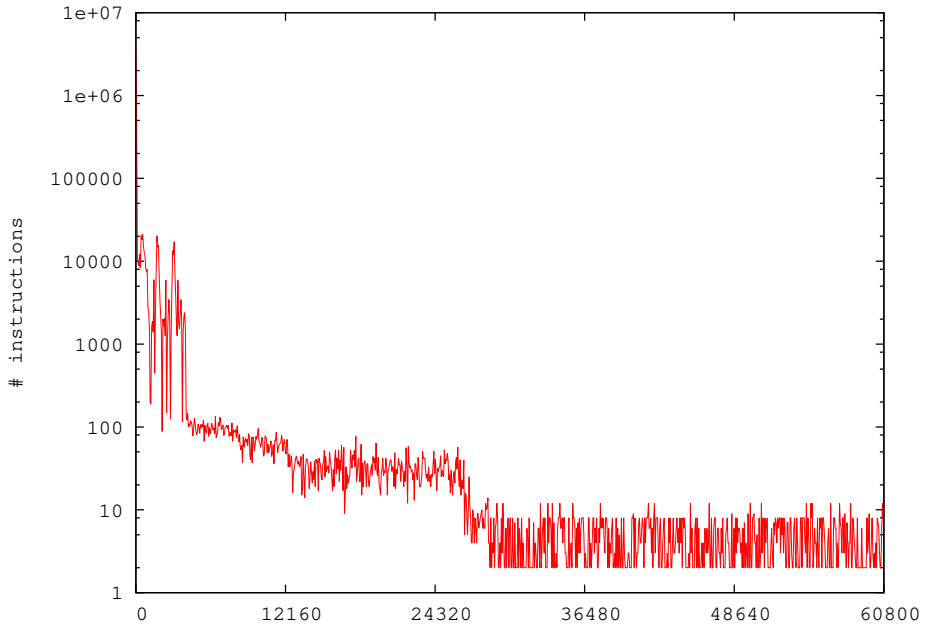
PerPI: Local vs Global ILP

Limited core / cjpeg (Mibench)



IPC = 3.1 (Local ILP)

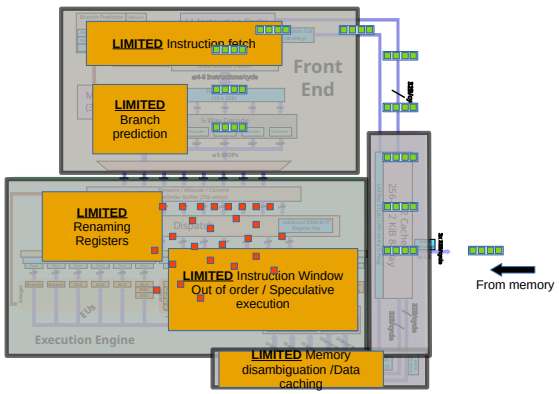
Unlimited core / cjpeg (Mibench)



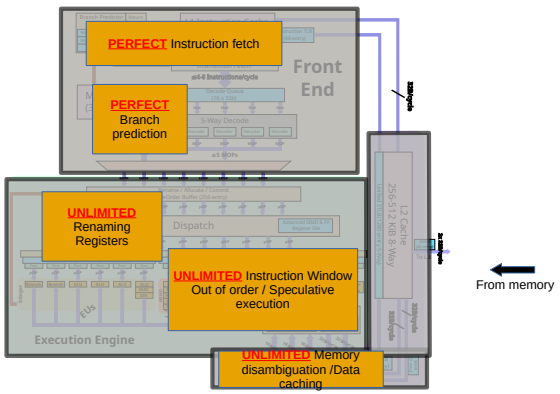
IPC = 1371 (Global ILP)

PerPI: an ILP Analyser (Core simulator)

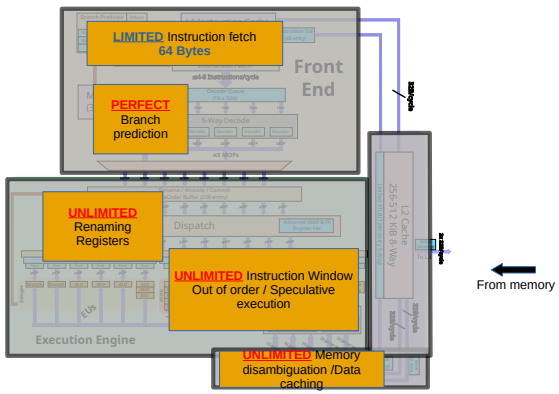
Limited core



Unlimited core

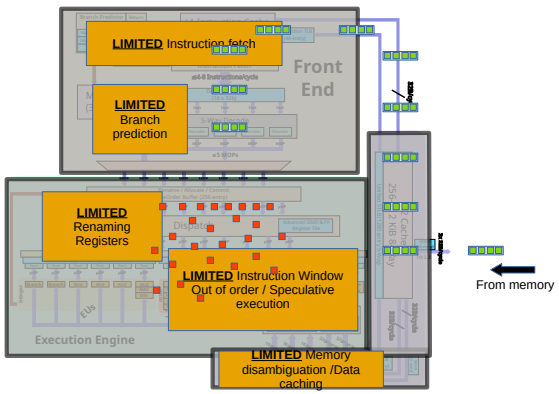


Quasi-unlimited core

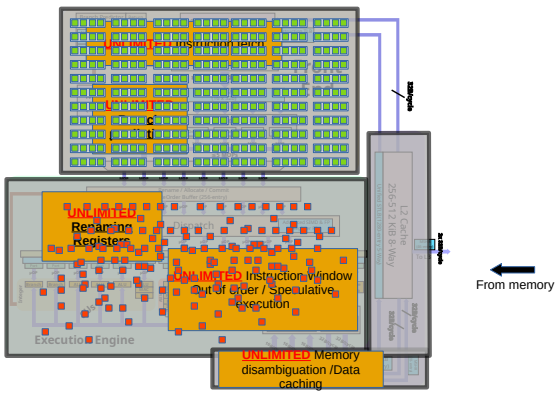


PerPI: an ILP Analyser (Core simulator)

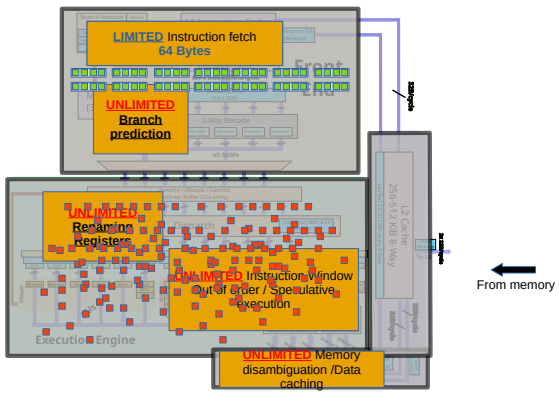
Limited core



Unlimited core

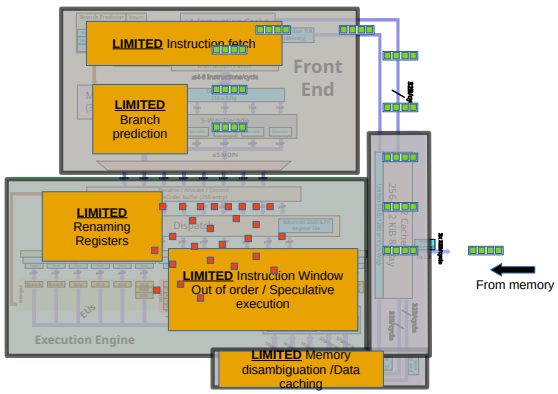


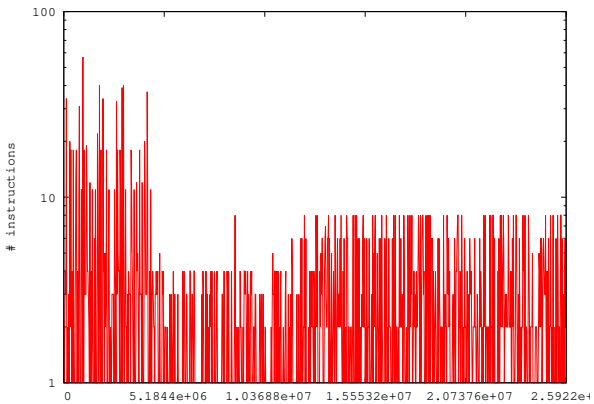
Quasi-unlimited core



PerPI: an ILP Analyser (Core simulator)

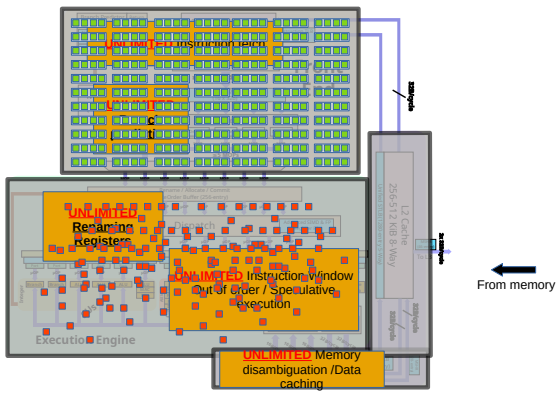
Limited core

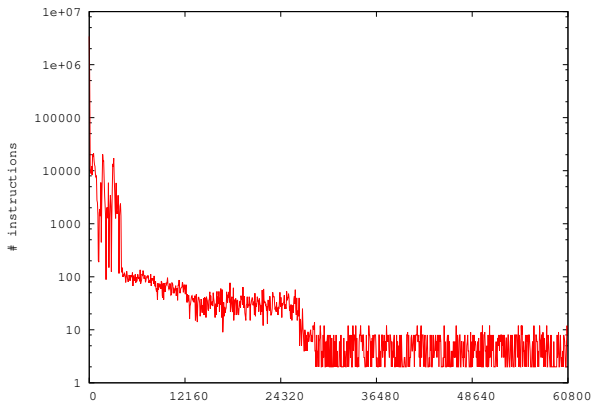




IPC = 3.1 (Local ILP)

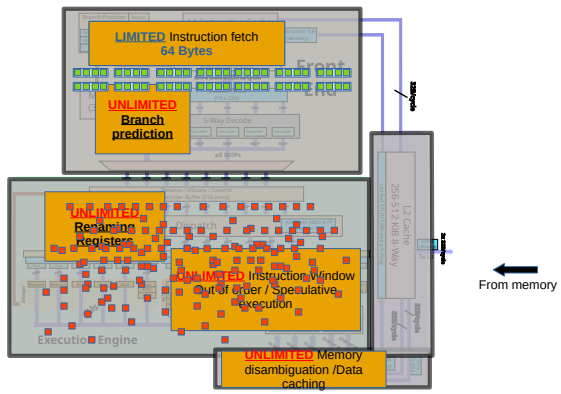
Unlimited core

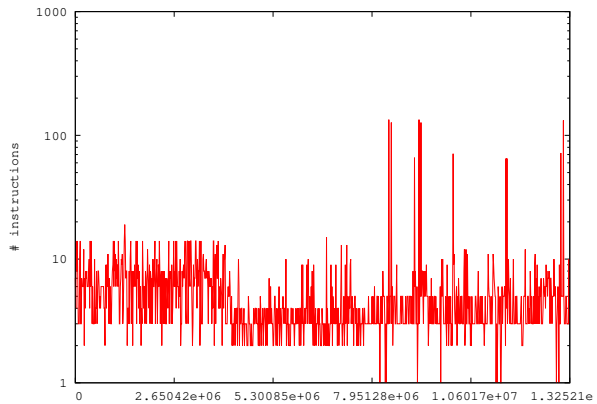




IPC = 1371 (Global ILP)

Quasi-unlimited core

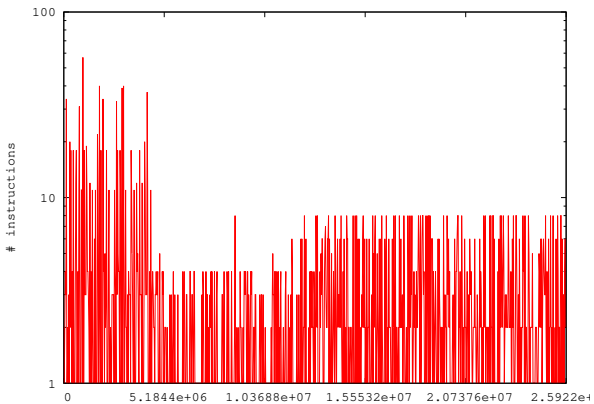




IPC = 6.3 (Local ILP)

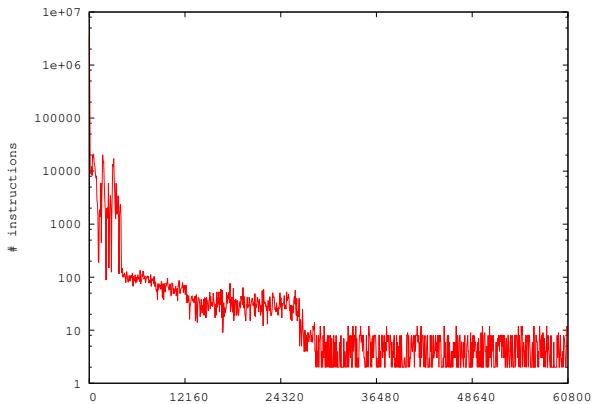
PerPI: Local vs Global ILP

Limited core / cjpeg (Mibench)



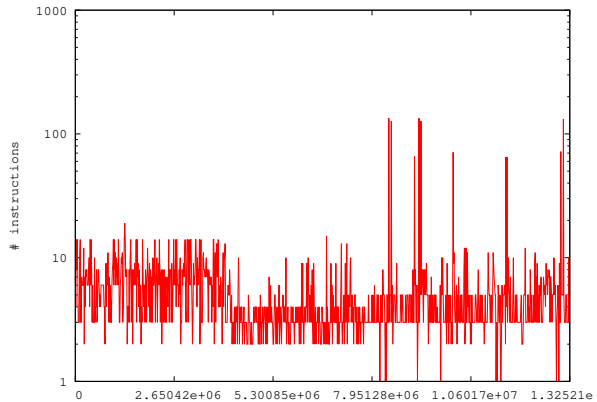
IPC = 3.1 (Local ILP)

Unlimited core / cjpeg (Mibench)



IPC = 1371 (Global ILP)

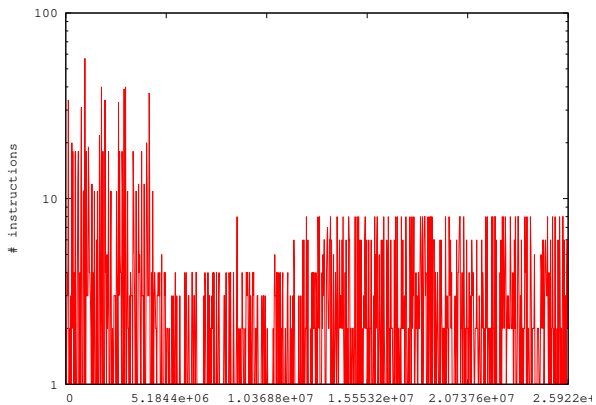
Quasi-unlimited core / cjpeg (Mibench)



IPC = 6.3 (Local ILP)

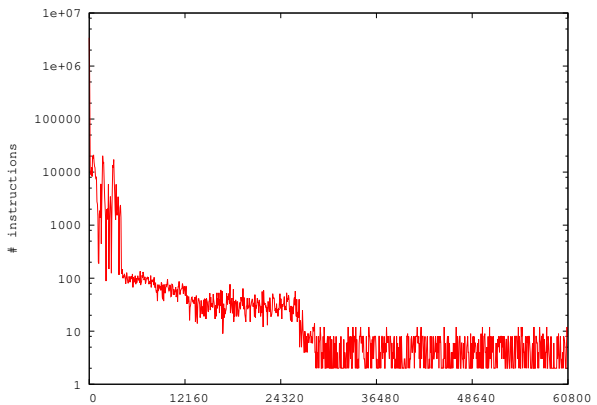
PerPI: Local vs Global ILP

Limited core / cjpeg (Mibench)



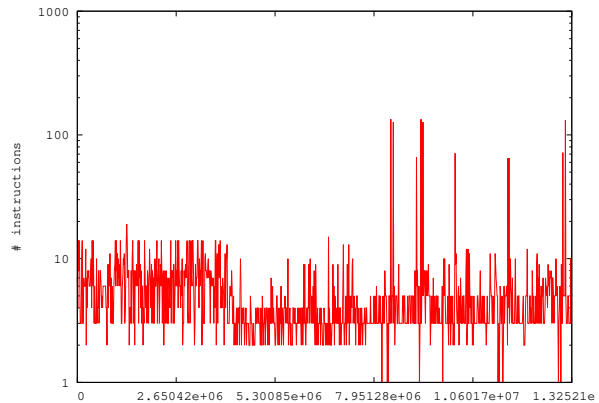
IPC = 3.1 (Local ILP)

Unlimited core / cjpeg (Mibench)



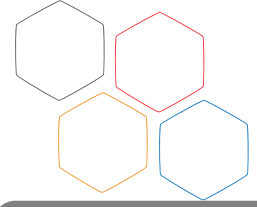
IPC = 1371 (Global ILP)

Quasi-unlimited core / cjpeg (Mibench)



IPC = 6.3 (Local ILP)

- Huge distance between independent instructions
- Enable register and memory renaming



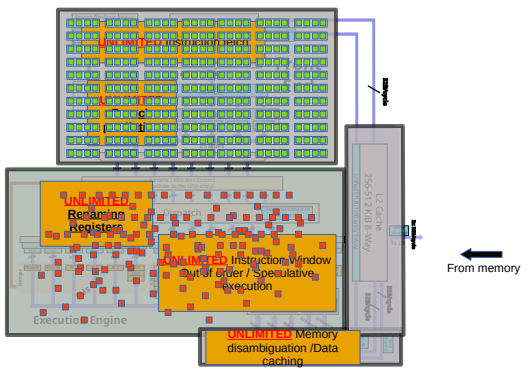
Limited, Unlimited, Quasi-unlimited

IPC of 3 the simulated cores

bench.	Limited	Unlimited	Quasi-unlimited
basicmath	3.08	104.78	5.60
bitcount	3.45	571.92	16.82
cjpeg	3.22	1371.41	6.29
crc	2.58	24.51	6.12
dijkstra	1.64	1297.20	4.66
djpeg	6.39	127.48	6.39
fft	3.26	194.36	5.54
ffti	3.16	156.37	5.03
lout	3.90	343.78	5.65
mad	6.67	31.39	7.39
patricia	3.04	174.30	5.19
qsort	2.87	586.34	5.47
rawc	3.78	8.21	6.56
rawd	4.62	7.71	7.71
say	3.59	96.70	3.59
search	2.66	254.73	7.87
susanc	3.47	6896.87	6.36
susane	3.32	6935.47	6.20
susans	6.01	1298921.27	57.79
toast	4.23	99.50	7.04
untoast	1.72	33.55	6.03

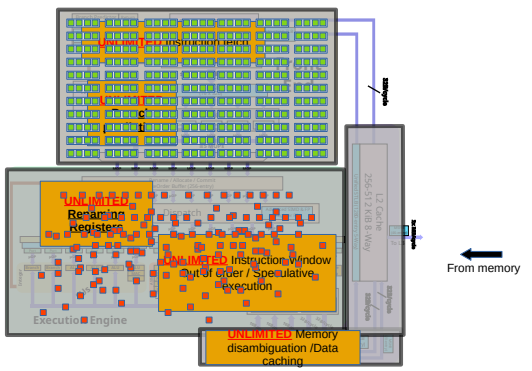
In the real world

Unlimited core (Quasi-impossible)

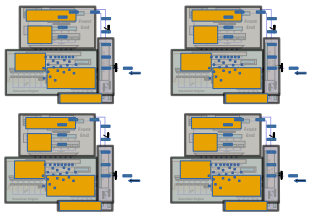


In the real world

Unlimited core (Quasi-impossible)

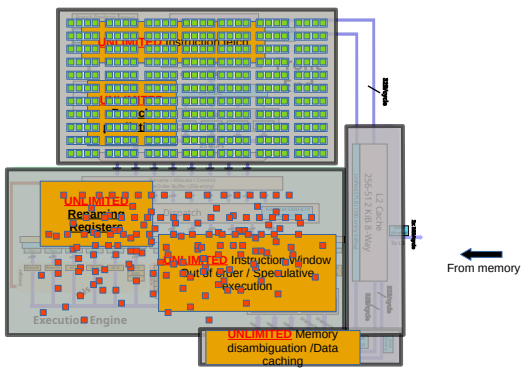


Parallel programming (Possible?)

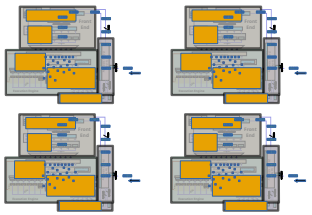


In the real world

Unlimited core (Quasi-impossible)

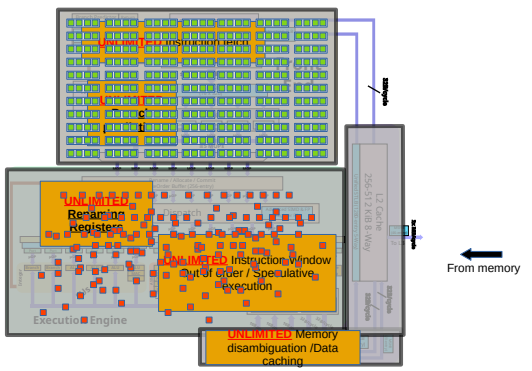


Parallel programming (Possible?)

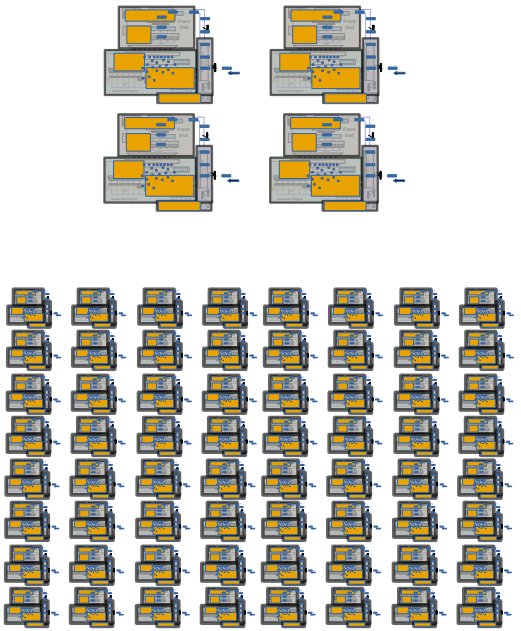


In the real world

Unlimited core (Quasi-impossible)



Parallel programming (Possible?)



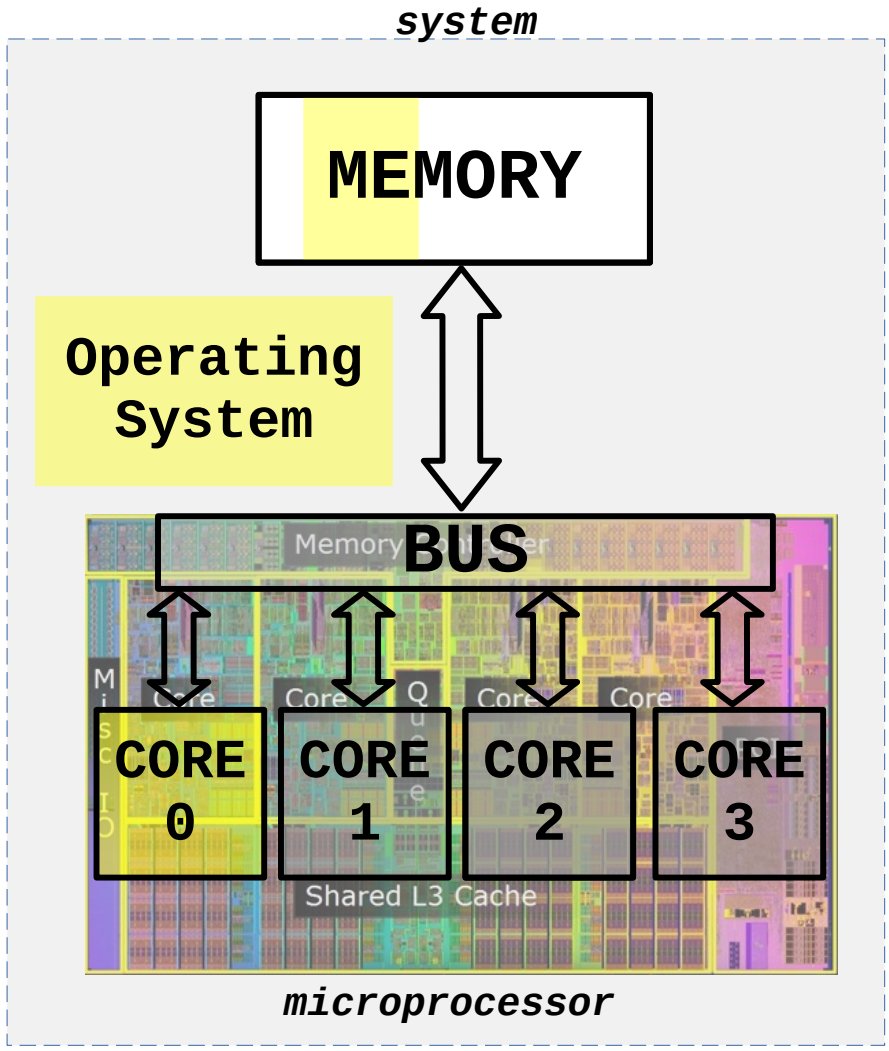
- multiple “threads” => multiple starting points
- multiple cores => register renaming
- “thread” private memory => memory renaming
- manycores + good parallel programming => Fake an “quasi-one-shot” access to the all execution trace

OpenMP execution (w/ OS threads)

```
void func(int i, int j)
{

   for (i=0;i<j;++i)
   {
      dosomething()
   }
}
```

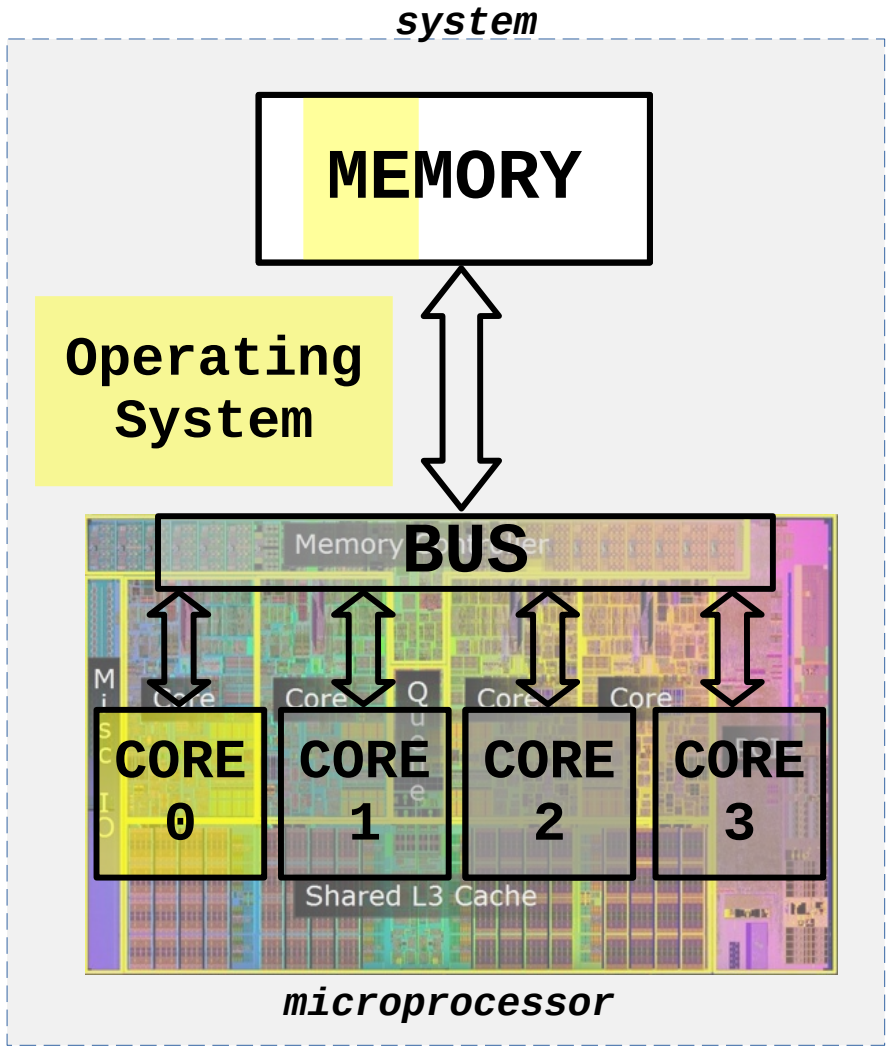
source



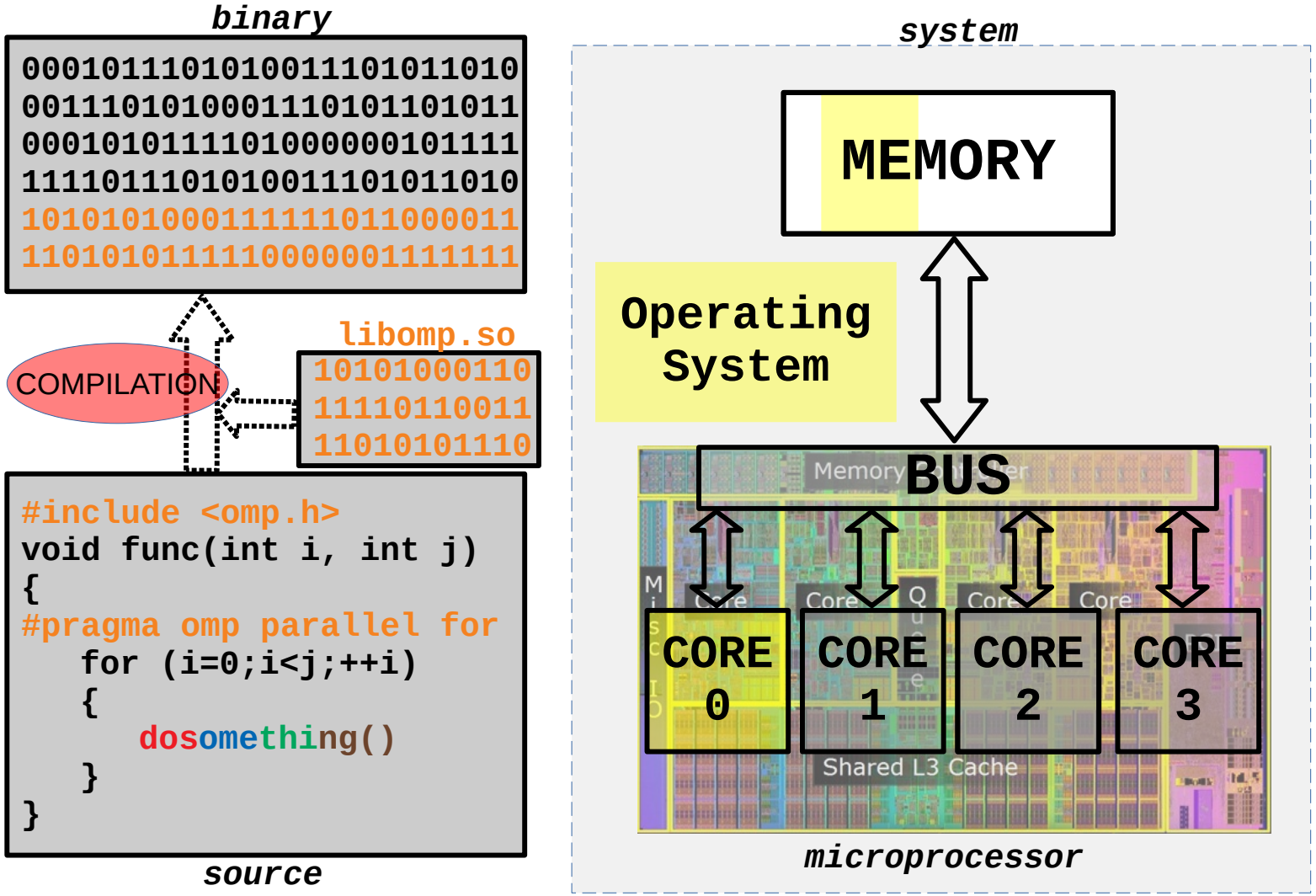
OpenMP execution (w/ OS threads)

```
#include <omp.h>
void func(int i, int j)
{
#pragma omp parallel for
   for (i=0;i<j;++i)
   {
      dosomething()
   }
}
```

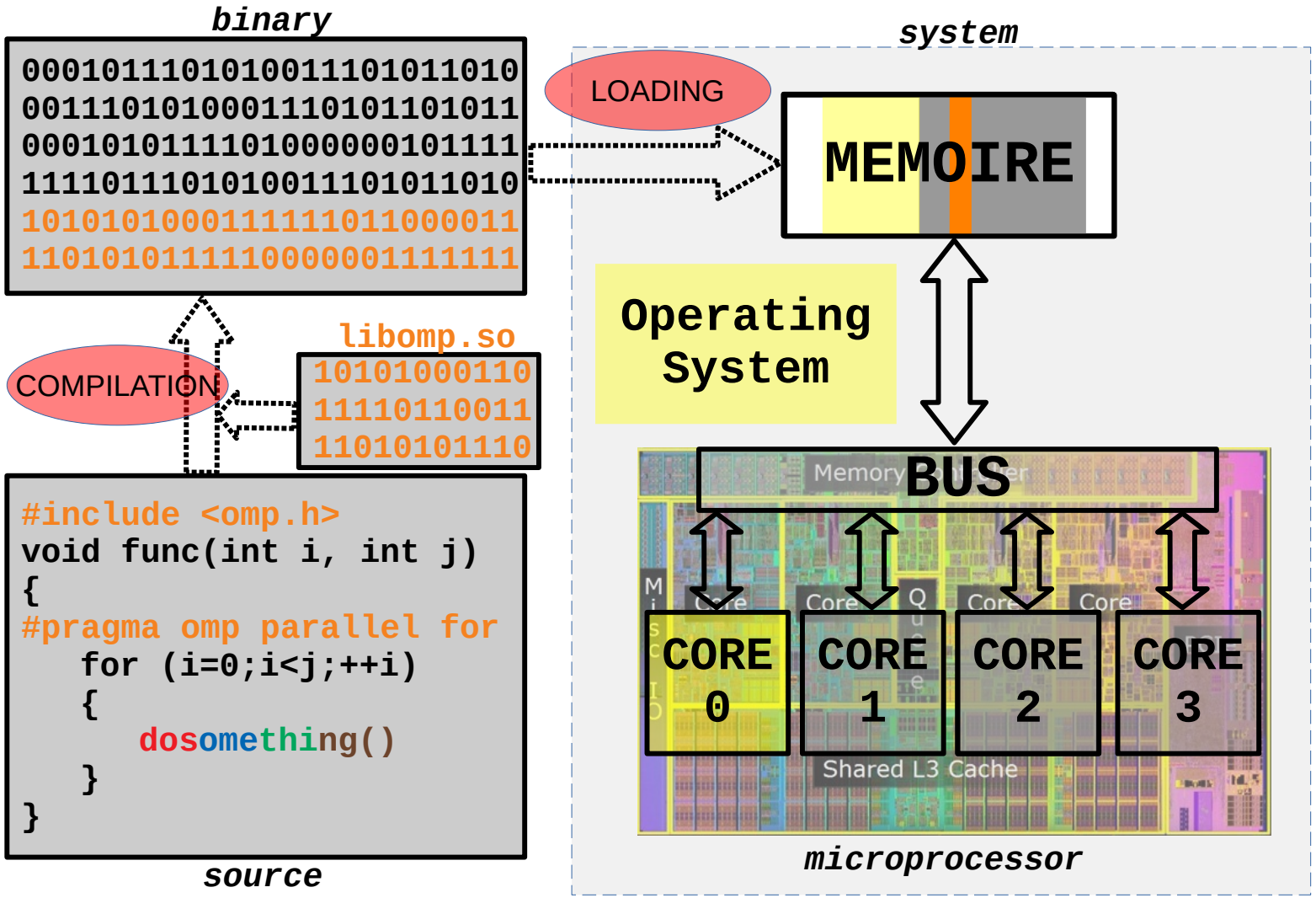
source



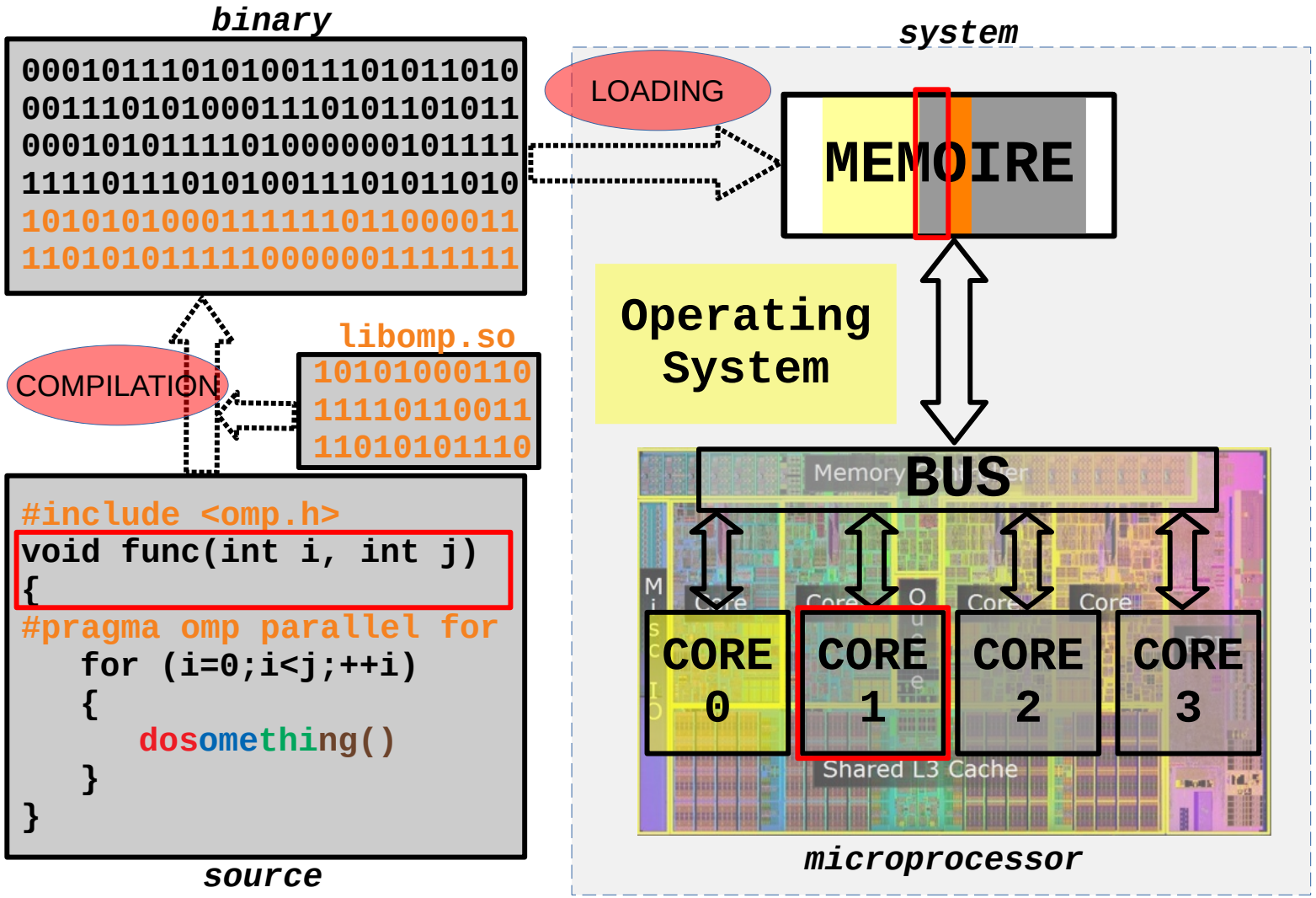
OpenMP execution (w/ OS threads)



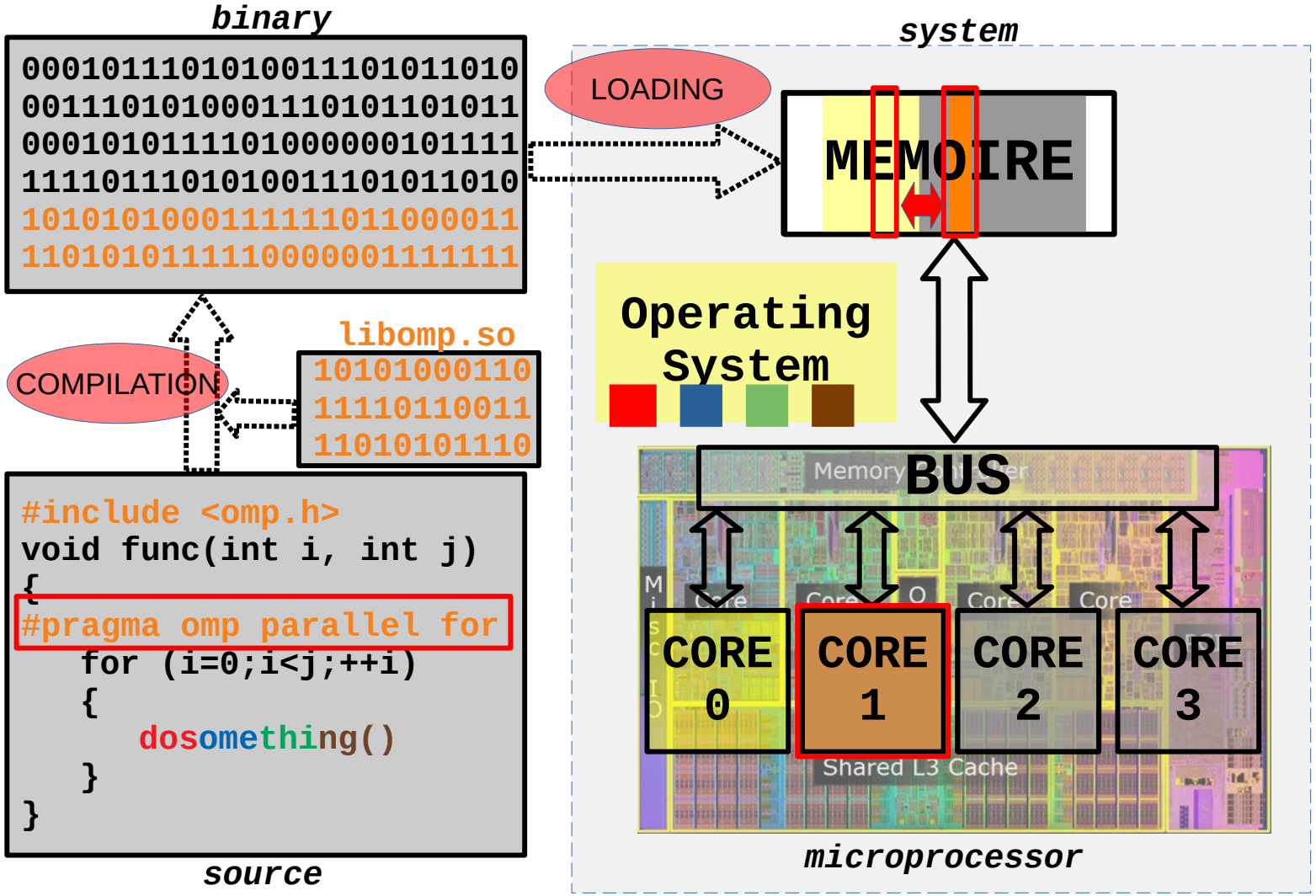
OpenMP execution (w/ OS threads)



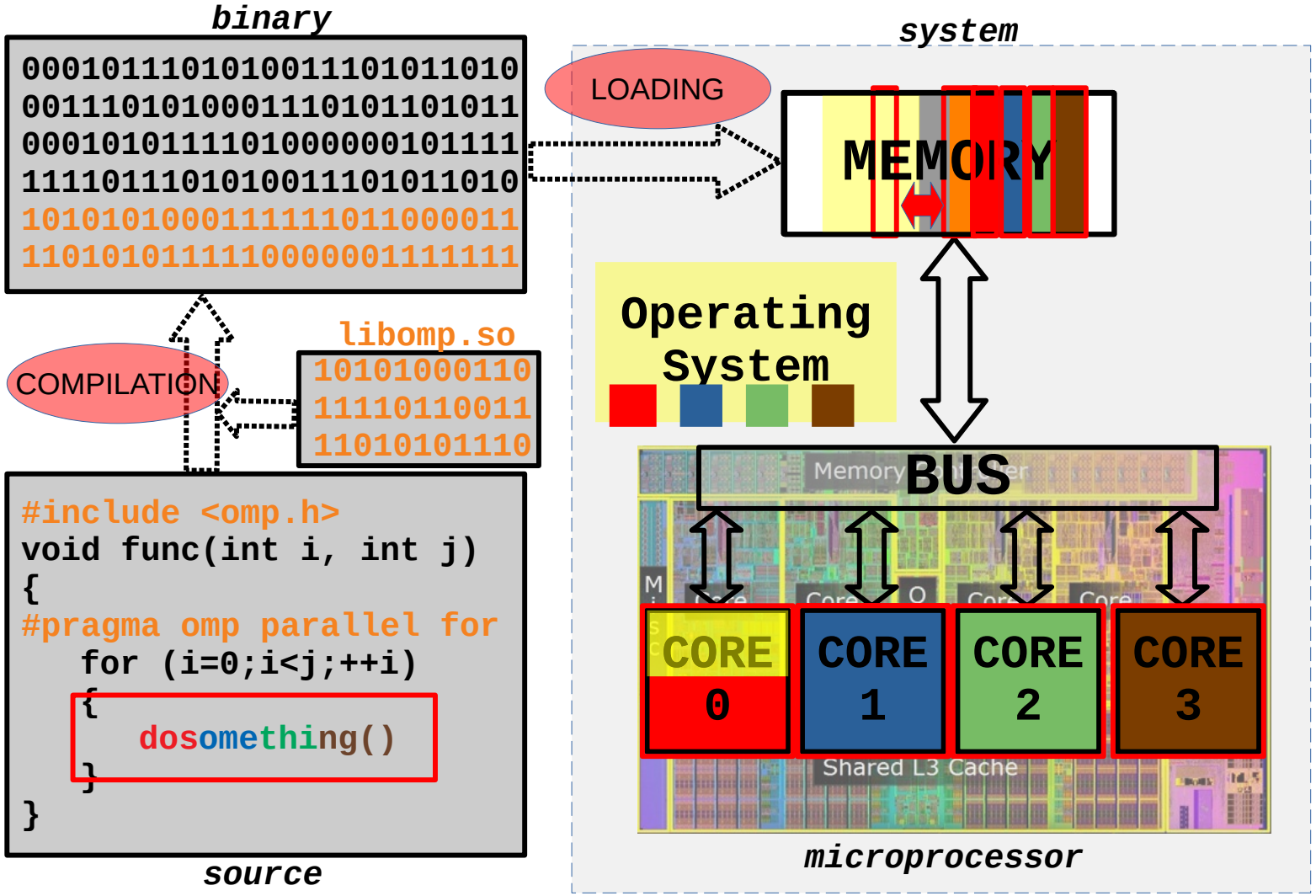
OpenMP execution (w/ OS threads)



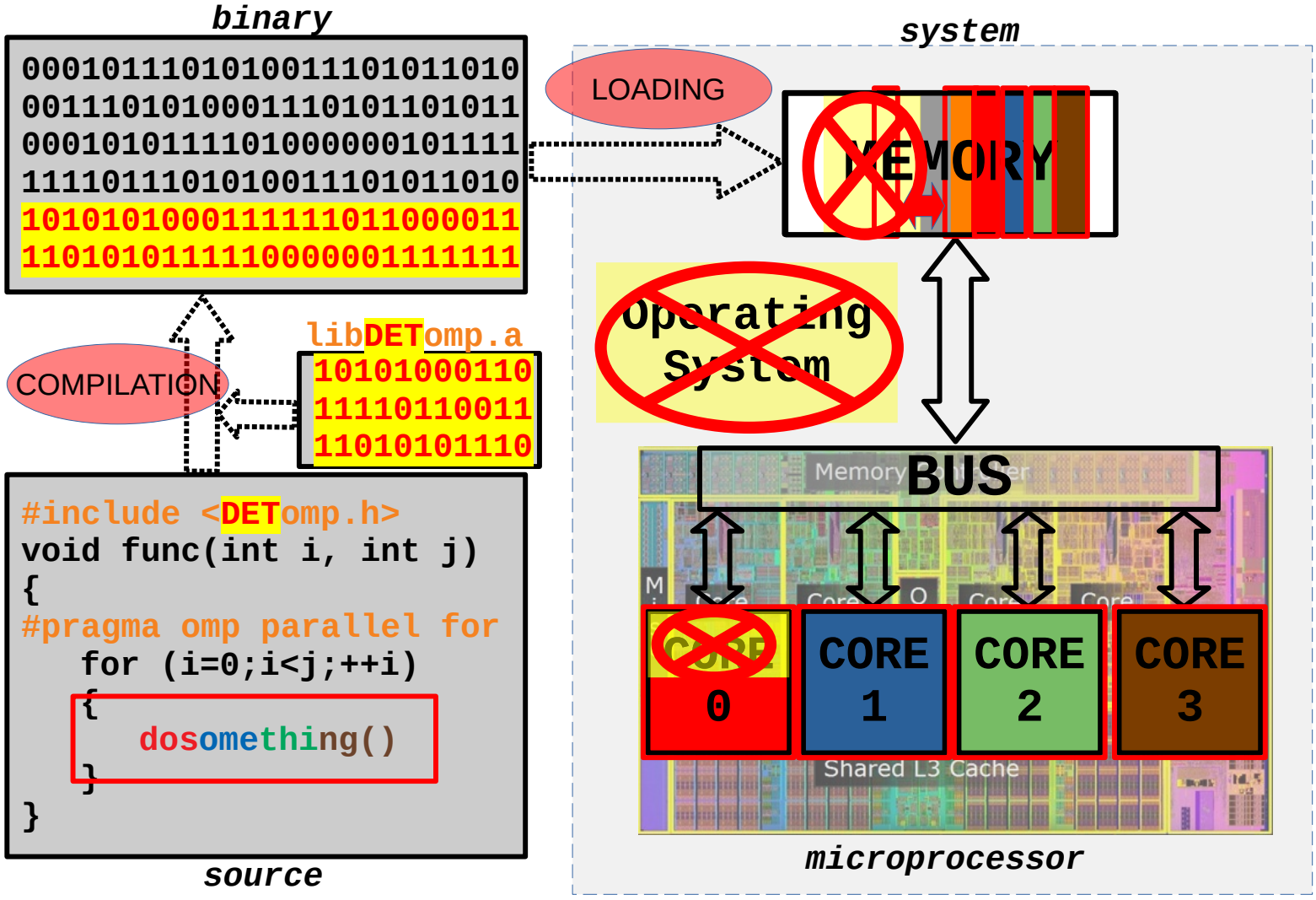
OpenMP execution (w/ OS threads)



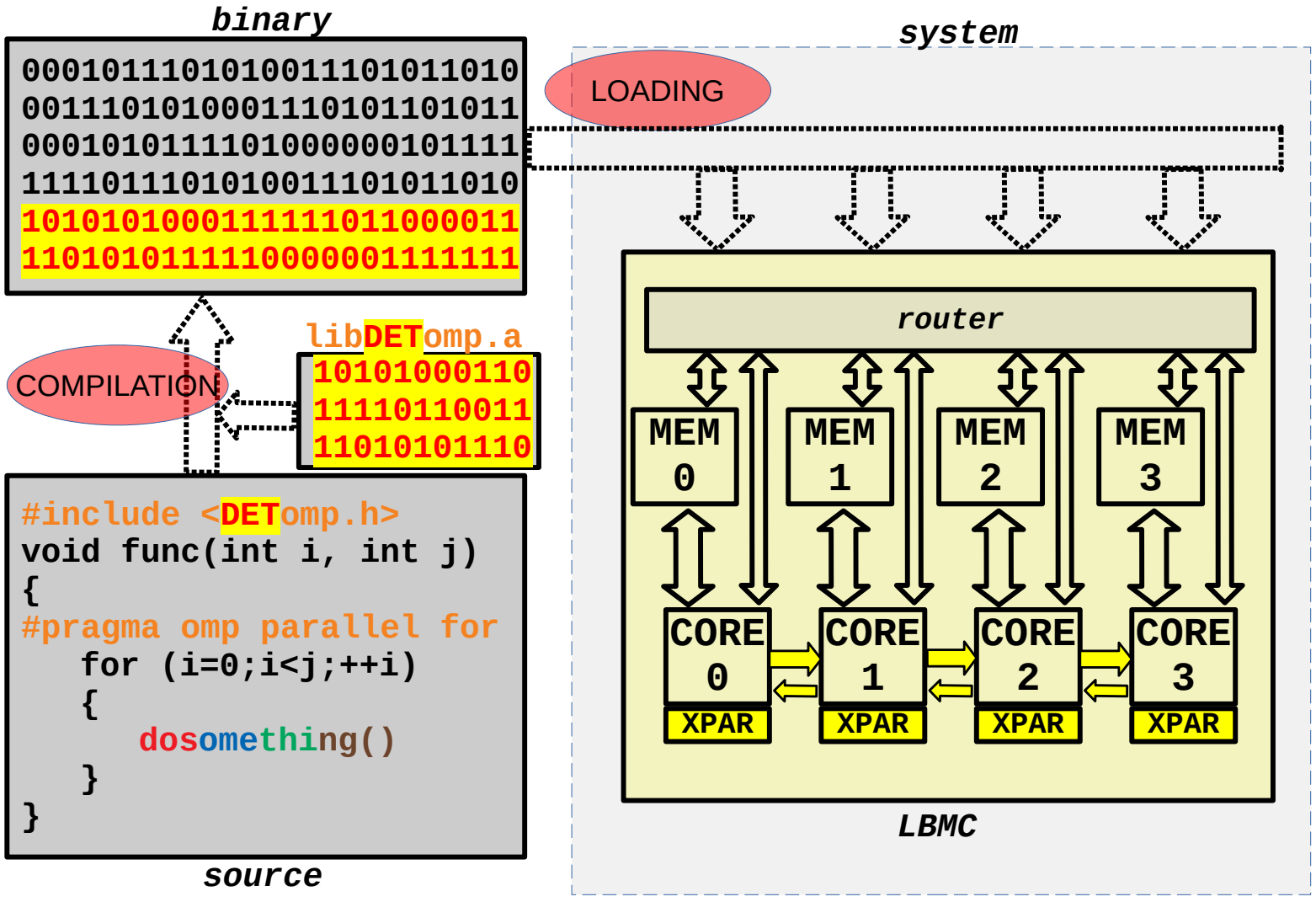
OpenMP execution (w/ OS threads)



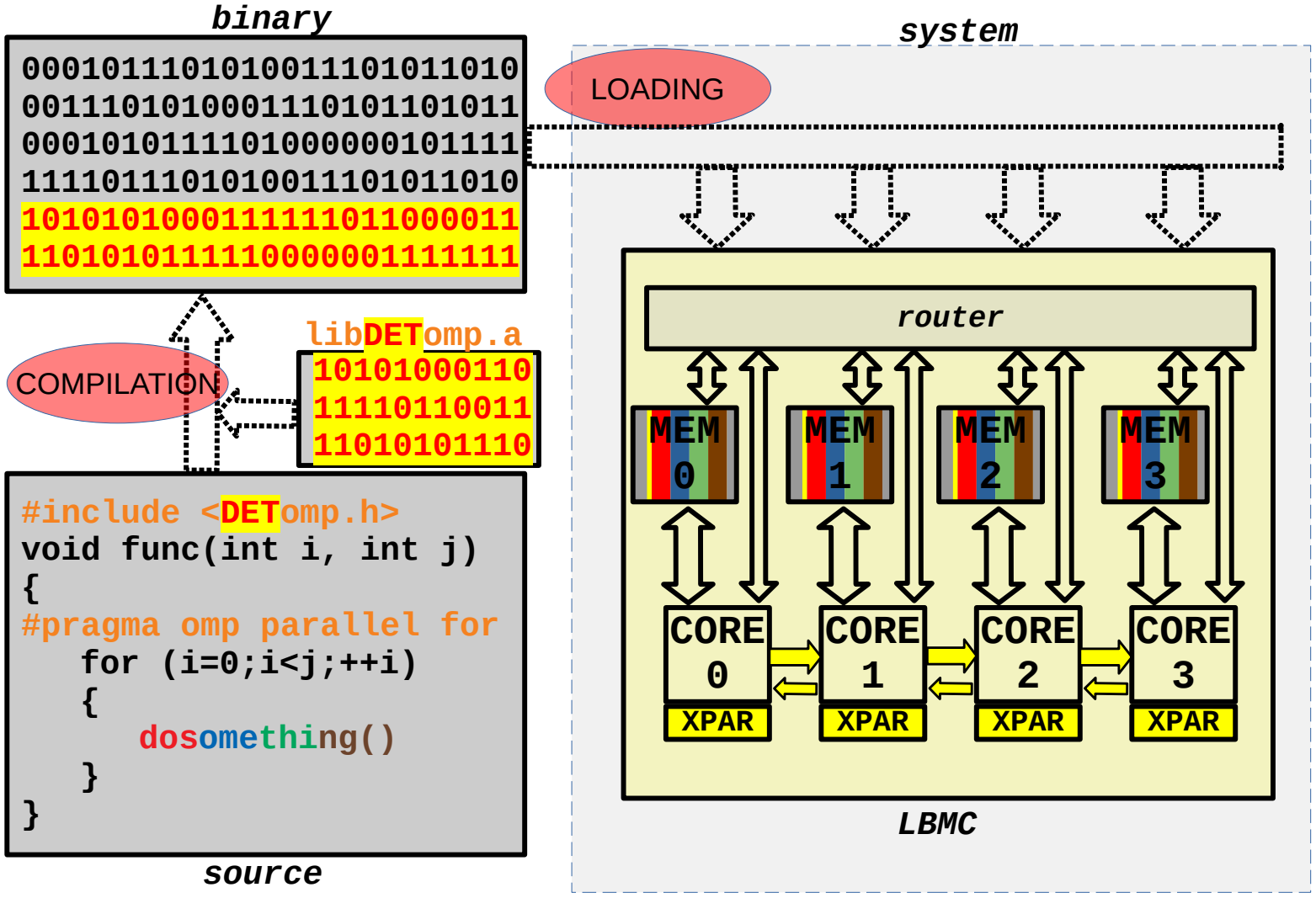
DETOpenMP execution (w/ Harts)



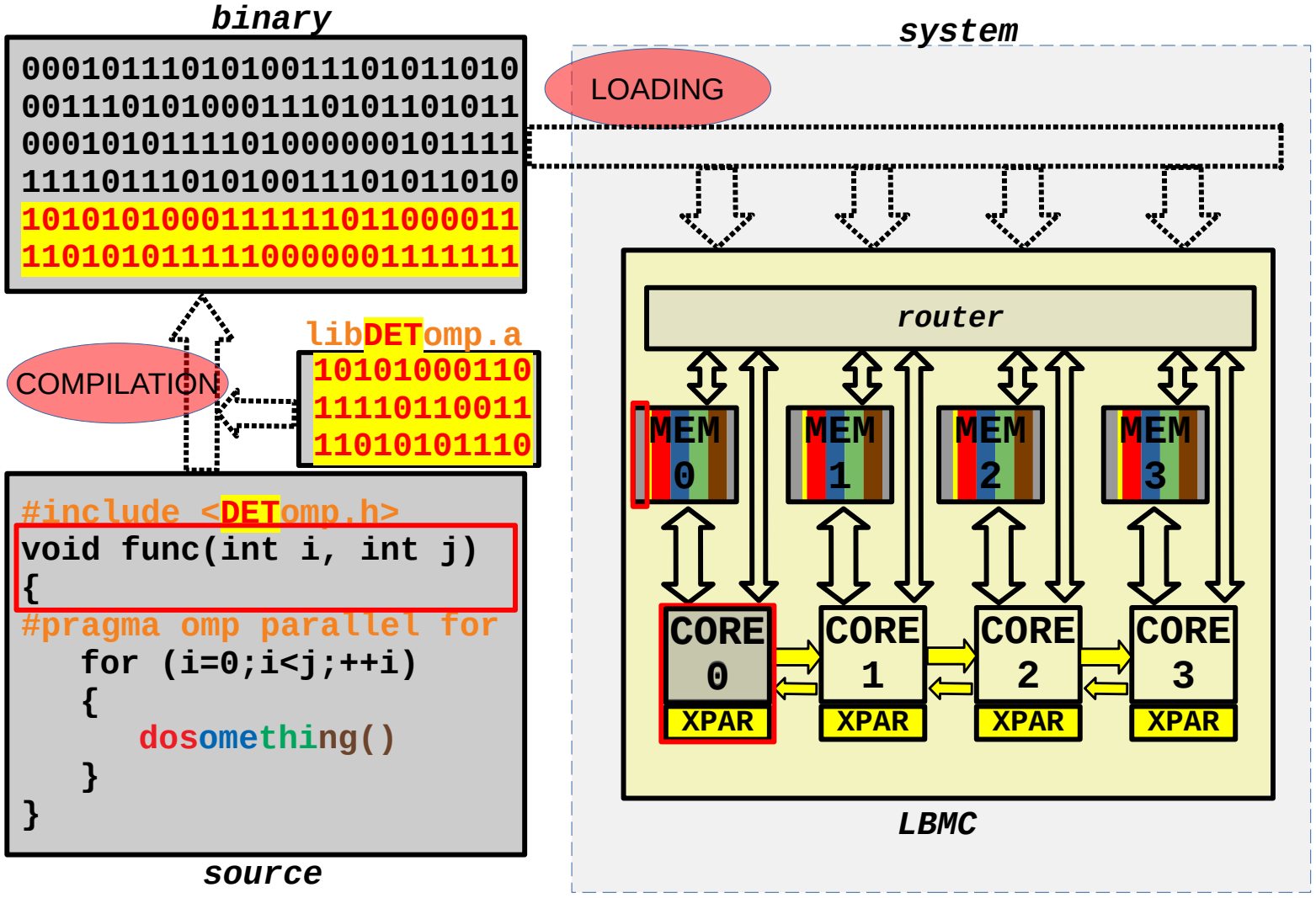
DETOpenMP execution (w/ Harts)



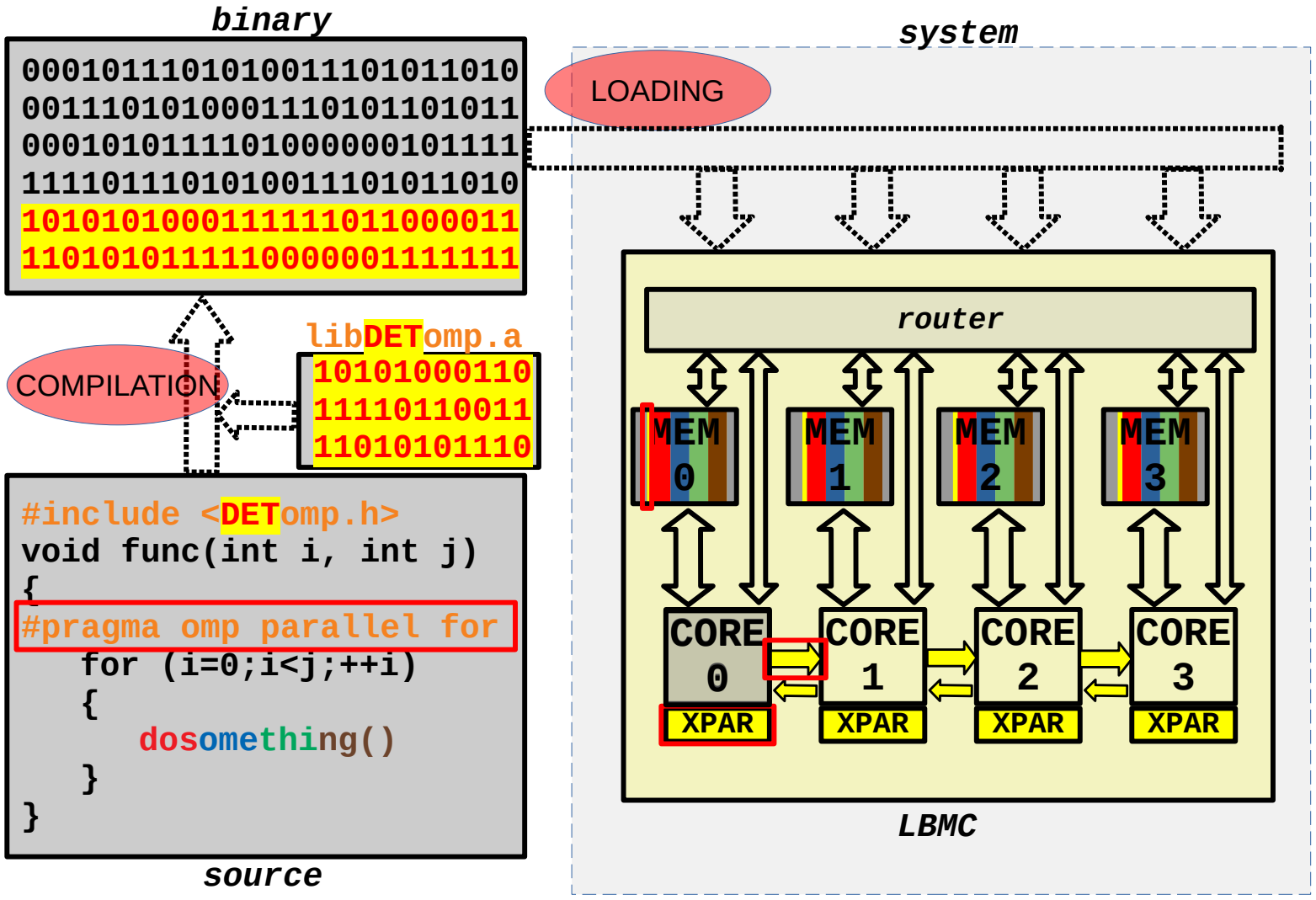
DETOpenMP execution (w/ Harts)



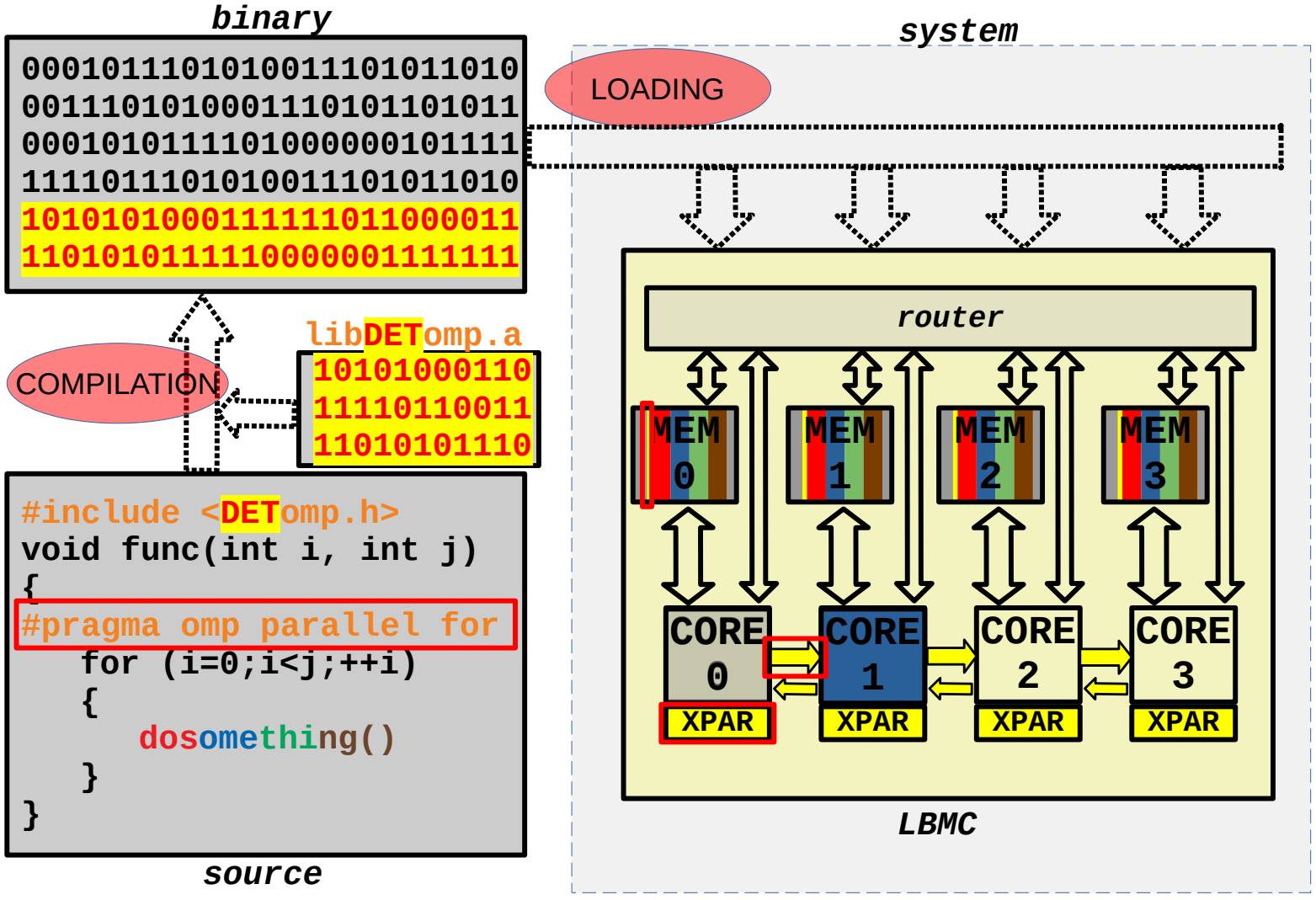
DETOpenMP execution (w/ Harts)



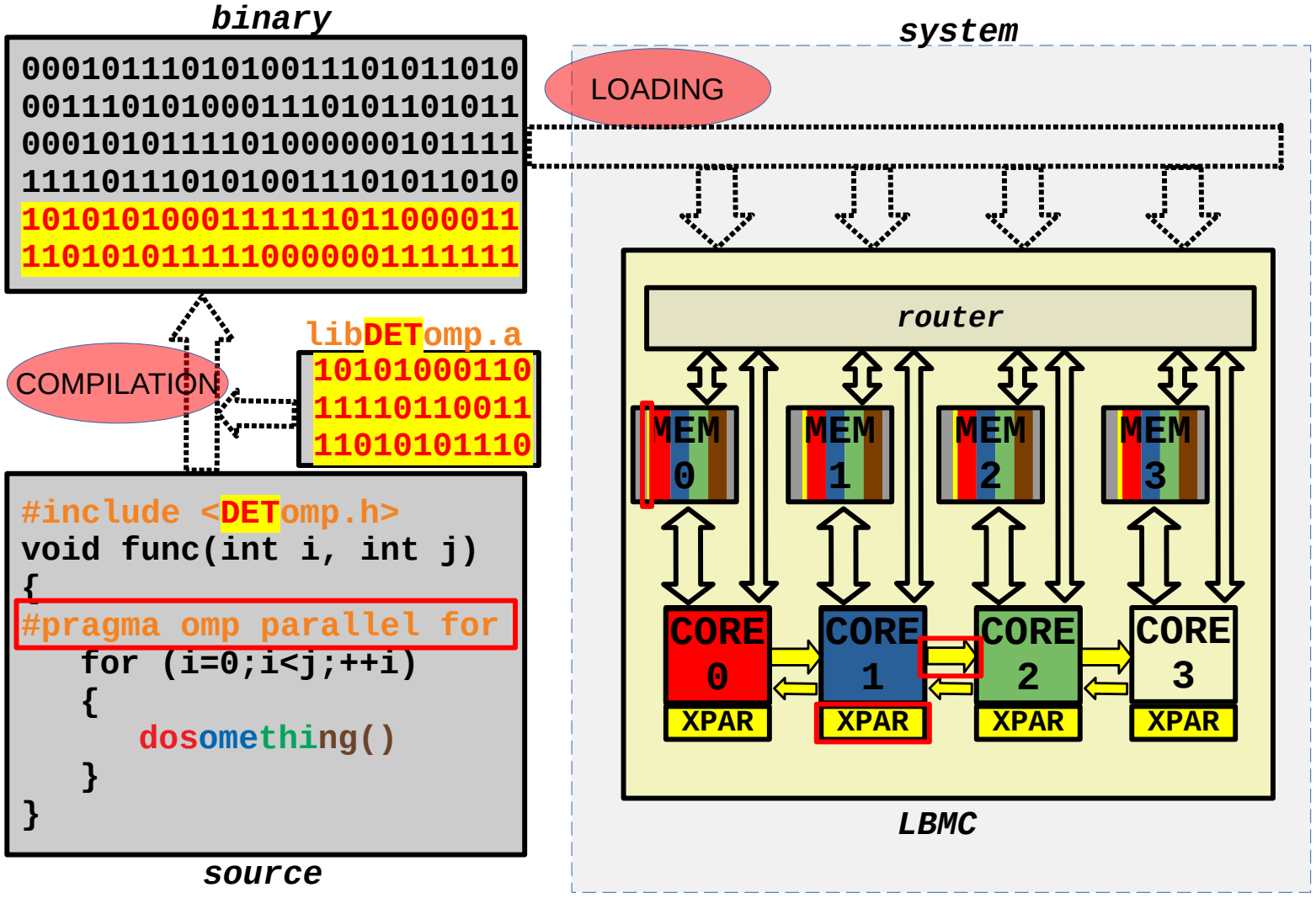
DETOpenMP execution (w/ Harts)



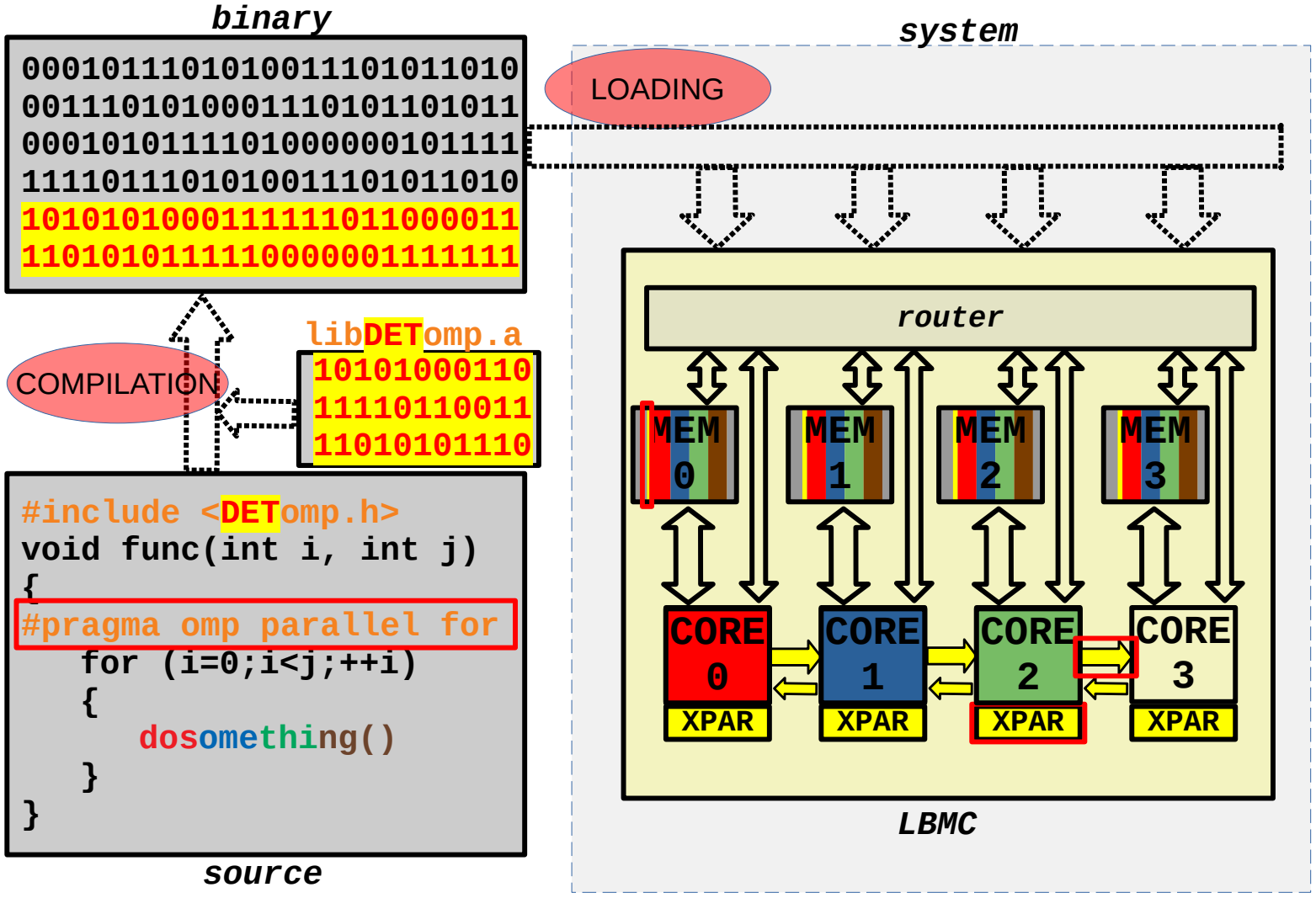
DETOpenMP execution (w/ Harts)



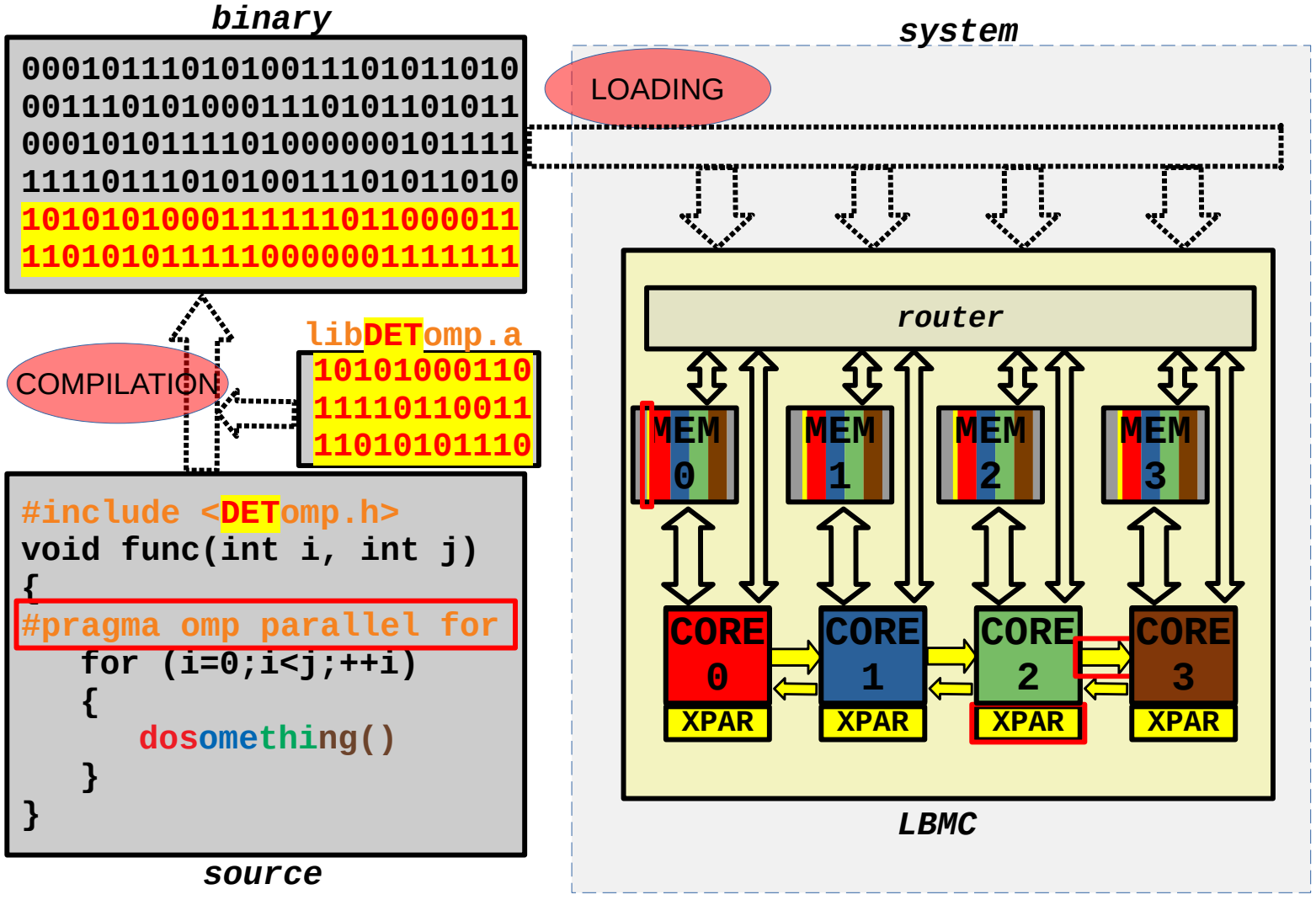
DETOpenMP execution (w/ Harts)



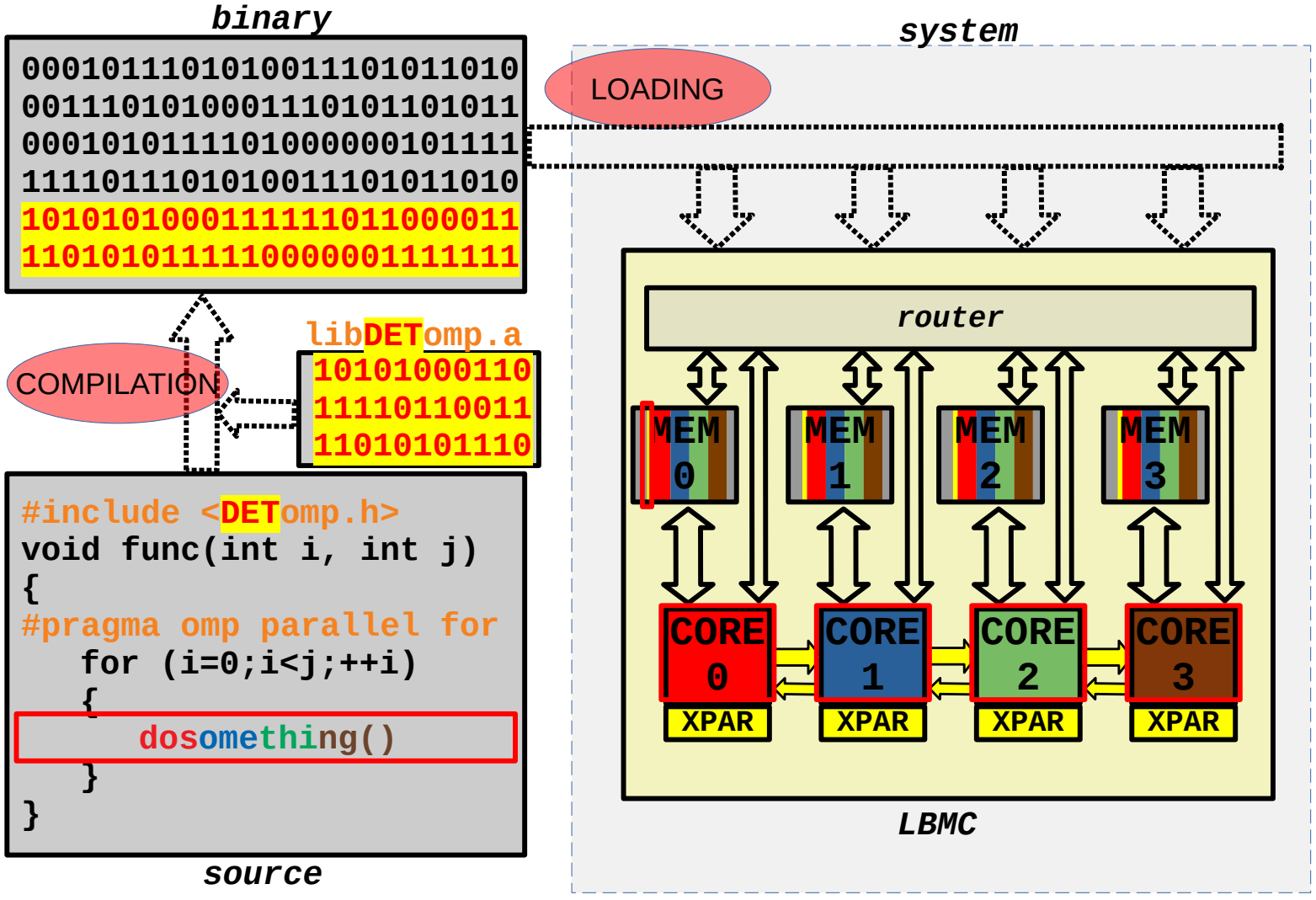
DETOpenMP execution (w/ Harts)



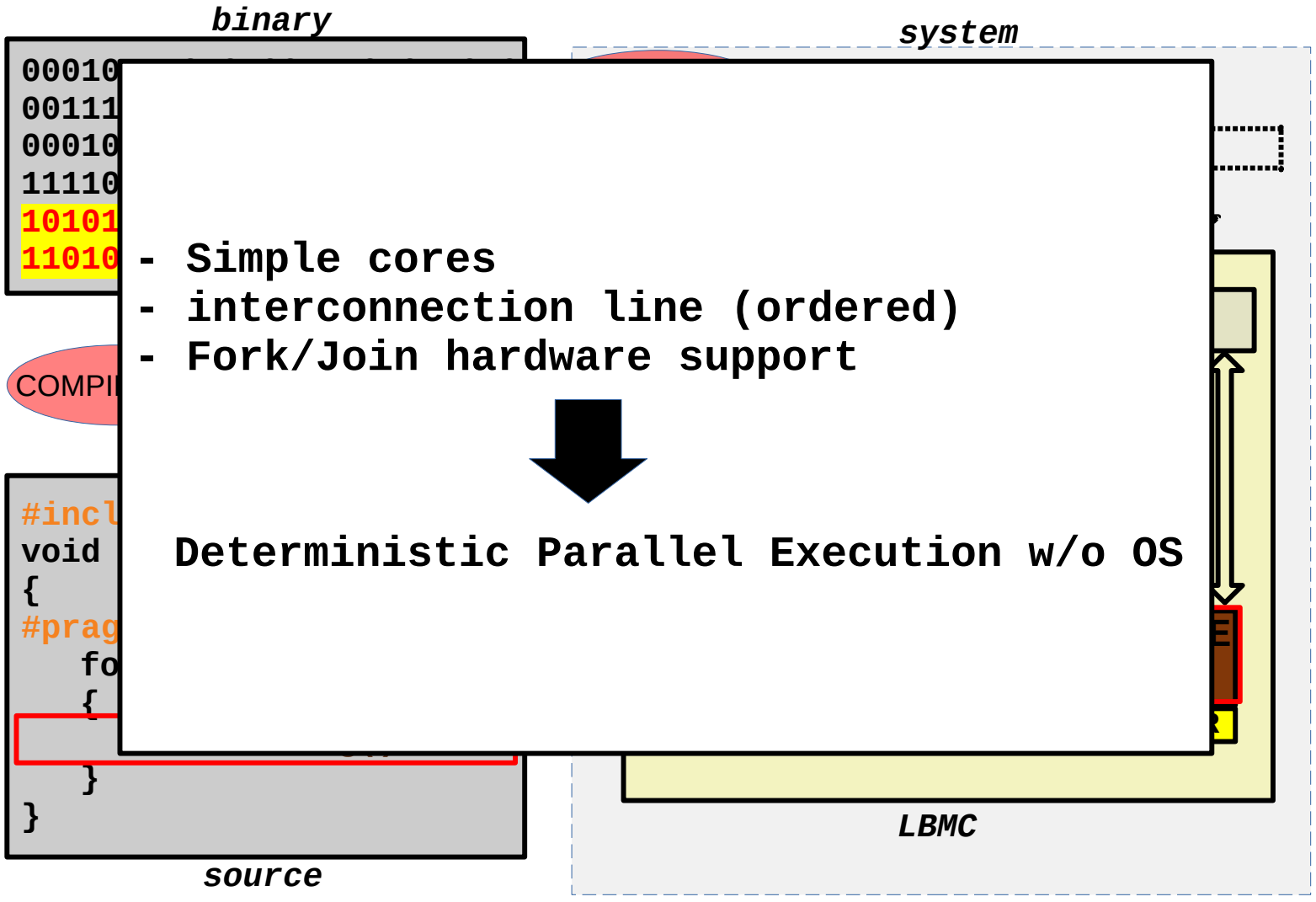
DETOpenMP execution (w/ Harts)



DETOpenMP execution (w/ Harts)



DETOpenMP execution (w/ Harts)

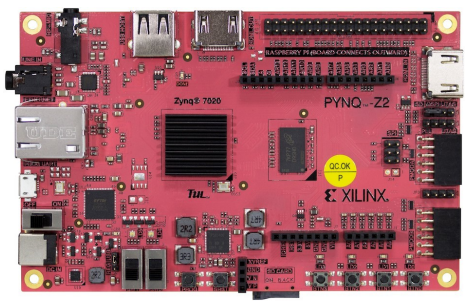


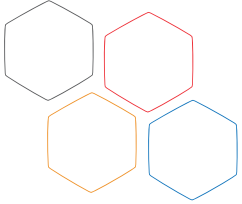
LBMC Prototype (RISCV 32 bits)

Very simple RV32IM cores

- No branch prediction
- No cache
- In order 6-stage pipeline (F,D,I,E,M,W)
- multiple hardware threads per core (HARTS)
- Specific interconnect

FPGA implementation (HLS / Vivado / Vitis)

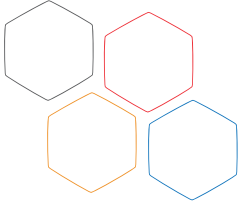




Matrix multiplication on LBMC

LBMC configurations

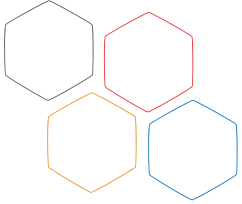
# of cores	# of harts	cycles	nmi	cpi	time(s)	speedup
1	1	6,2 M.	3,8 M.	1.62	0.125	-
2	2	3,0 M.	4,6 M.	1.34	0.061	2.03
2	4	2,4 M.	4,7 M.	1.04	0.049	2.53
*2	8	2,6 M.	5,0 M.	1.07	0.054	2.31
4	2	1,5 M.	4,7 M.	1.34	0.032	3.94
*4	4	1,3 M.	5,0 M.	1.05	0.027	4.70
*8	2	0,8 M.	5,0 M.	1.34	0.017	7.34



Conclusions

Conclusions

- Applications may have a high potential of ILP
- Good parallel programming skills and good tools
- Only use simple hardware to exploit Global ILP



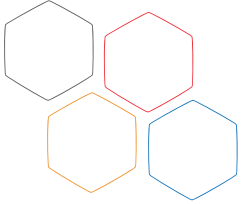
Conclusions

Conclusions

- Applications may have a high potential of ILP
- Good parallel programming skills and good tools
- Only use simple hardware to exploit Global ILP

LBMC prototype

- Extremely simple RV32IM cores
- Specific interconnect
- DETOpenMP library



Conclusions

Conclusions

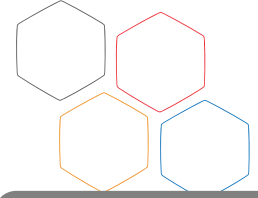
- Applications may have a high potential of ILP
- Good parallel programming skills and good tools
- Only use simple hardware to exploit Global ILP

LBMC prototype

- Extremely simple RV32IM cores
- Specific interconnect
- DETOpenMP library

Advantages

- LBMC run deterministic parallel programs w/o OS and w/o interruption
- Extremely simple core -> less vulnerabilities (side channel attacks)



Future works

On going works

- DetOpenMP library development (adding OMP pragmas)
- LBMC optimizations (ex: core size)

Future works

- Develop LBMC Debugger tool
- LBMC design space exploration

Future works

On going works

- DetOpenMP library development (adding OMP pragmas)
- LBMC optimizations (ex: core size)

Future works

- Develop LBMC Debugger tool
- LBMC design space exploration
- Explore multi-LBMC ?