Compiler technqiues to improve ILP

- Static scheduling: Re-ordering
- Loop unrolling

M[0+M]

Reduce stalls

LWD a, 0(M) IF ID EX M W

fpadd d, a, b IF ID - EX1

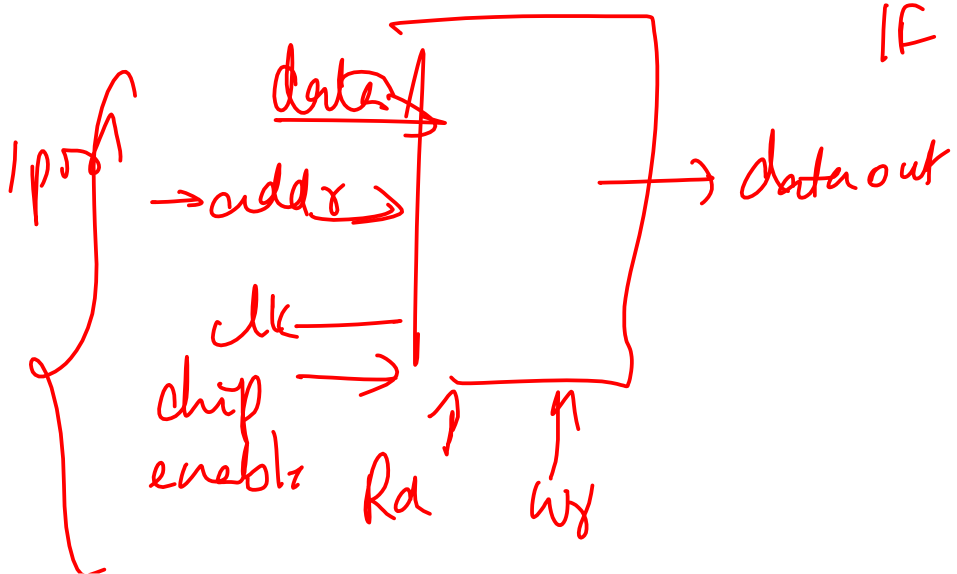
Single ported

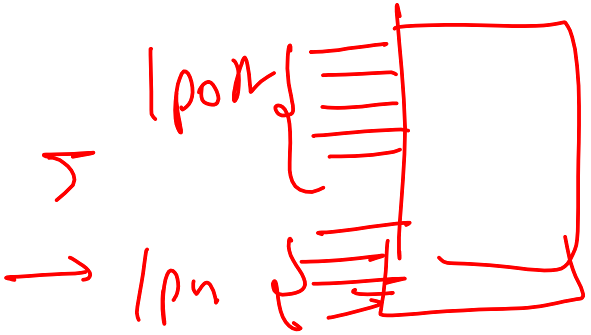
Dual ported.

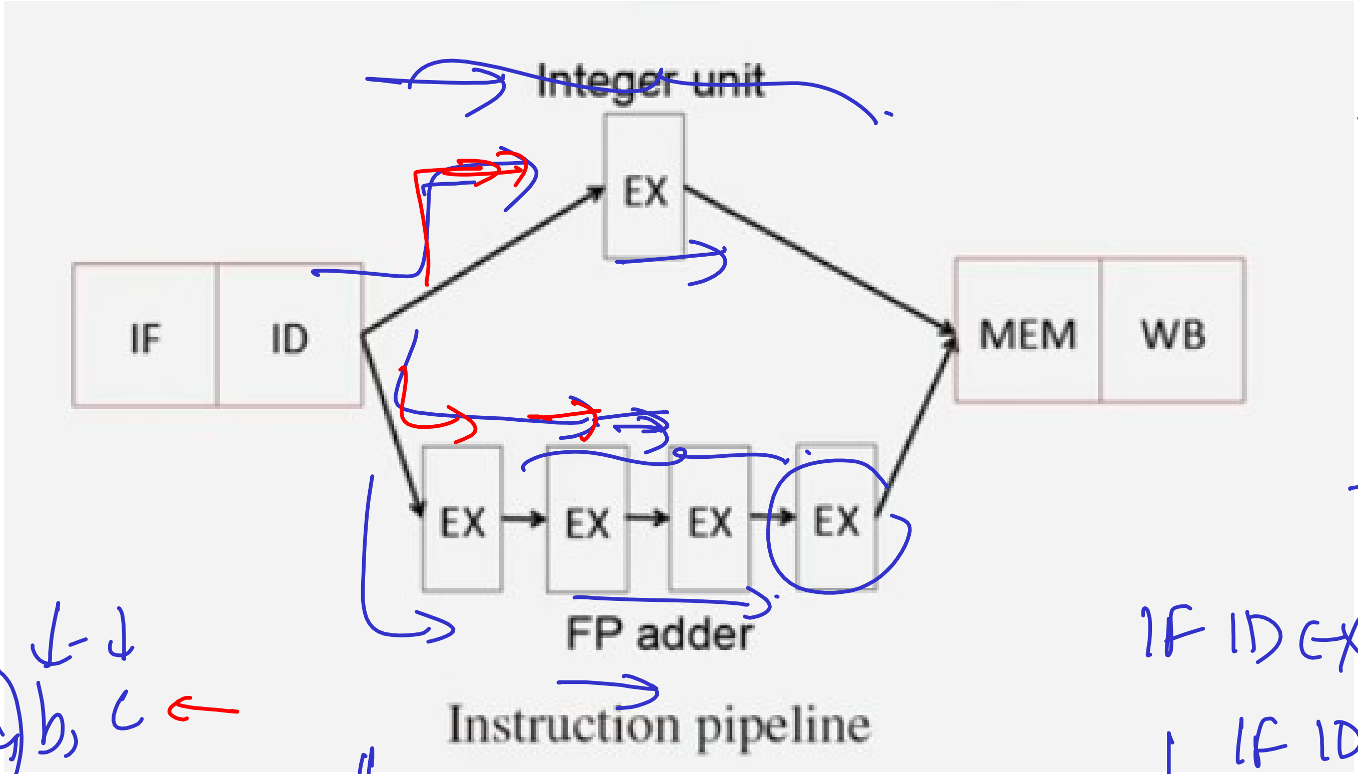
IF ID EX1 EX2 3 $\boxed{4}$ M W

IF ID EX - - - M

0+M







Instruction pipeline

Diverse pipeline

→ Int

IF ID EX1 2 3 4 | M W

IF ID – – – EX1 (4)

F

a,

FP MUL a, b, c ←

SW double a, 0, (M)

0+M

Value Producing instruction	Consuming instruction	Pipeline Latency/Number of stalls
FP ALU op	Another FP ALU op	3
FP ALU op (path2)	Store double (path 1)	2 + 1 for structural hazard
Load double	FP ALU op	1
Load double	Store double	0

Loop unrolling

```
for (i=999; i>=0; i --)
⇒   x[i] = x[i] + y;
```

Operation on floating point doubles--> 8 bytes of space in memory

Loop unrolling

memory.

```
for (i=999; i>=0; i --)
   x[i] = x[i] + y;
```

compile

x[i]

M[0+R1]

```
Loop:
L.D   F0, 0(R1)  // i = R1 = integer reg. F0 = floating pt reg
FADD.D  F4, F0, F2   //y =  F2
S.D  F4, 0(R1)
SUBI   R1, R1, #8
BNEQ  R1,R2 Loop  //R2=0
```

i

x(i) + y

i--

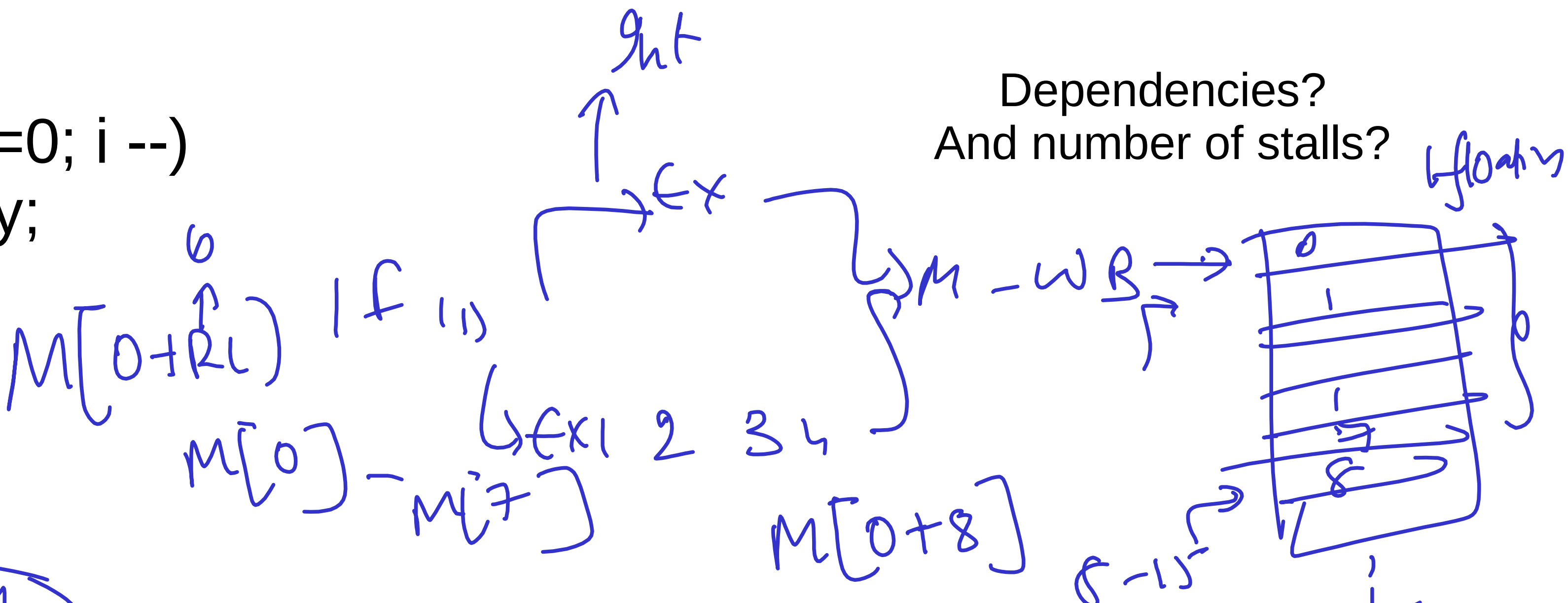
Dependencies?

Loop unrolling

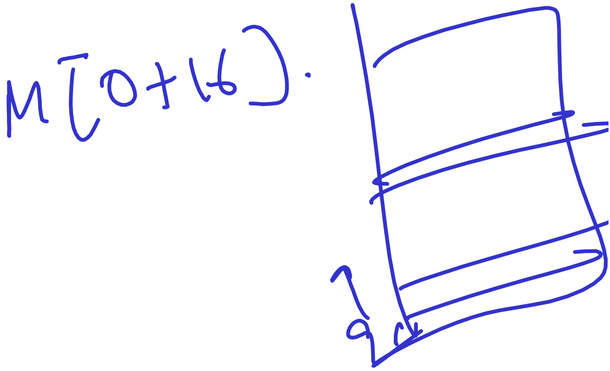
In-order WB.

```
for (i=999; i>=0; i --)
   x[i] = x[i] + y;
```

Dependencies?
And number of stalls?



```
Loop:
L.D    F0, 0(R1)  // i = R1 = integer reg. F0 = floating pt reg
ADD.D      F4, F0, F2    //y =  F2 – floating pt adder pipeline
S.D  F4, 0(R1)
SUBI    R1, R1, #8
BNEQ  R1,R2 Loop  //R2=0
```



Stalls

LD	IF	ID	EX	M	W					
Fadd		IF	ID	~~--~~	EX1	EX2	EX3	EX4	M	W
store			IF	~~IF~~ IF	ID	-- Ex	~~--~~ Ex	~~EX~~ Ey	~~--~~ Ex	M	W

ID

Structural hazard

IF ID

IF

Ex M W

ID Ex

IF

Loop:

L.D F0, 0(R1)

Stall

ADD.D F4, F0, F2 //y = F2

2 extra stalls +1 stall

due to structural hazard

3stall

SuB

BNEQ

Reorder?

S.D F4, 0(R1) (0+R1) - 9nt

SUBI R1, R1, #8

Stall --> fast branching for Branch / decide in ID

BNEQ R1,R2 Loop //R2=0

0 +R R1, R2

Ex

IF ID

LD ← ST

Add

SD → 3 ST

SuB ← 1 ST

BNEQ

Re-ordering/Scheduling

8

Loop:

L.D F0, 0(R1)

Stall

ADD.D F4, F0, F2 //y = F2

3 stalls

~~8~~ 0

S.D F4, 0(R1) S1

SUBI R1, R1, #8

R1=0

Stall

Add SD

BNEQ R1,R2 Loop

0

Loop:

L.D F0, 0(R1)

SUBI R1, R1, #8 R1=0

ADD.D F4, F0, F2

3 stalls

S.D F4, ~~0~~ 8(R1)

BNEQ R1,R2 Loop

5 * 1000 = 5000 instructions, 2 stalls

Re-ordering/Scheduling

```
Loop:
L.D   F0, 0(R1)
Stall
ADD.D     F4, F0, F2    //y =  F2
3 stalls
S.D  F4, 0(R1)
SUBI    R1, R1, #8
Stall
BNEQ  R1,R2 Loop
```

→

```
Loop:
L.D   F0, 0(R1)
SUBI    R1, R1, #8
ADD.D     F4, F0, F2
3 stalls
S.D  F4, 8(R1)
BNEQ  R1,R2 Loop
```

5 * 1000 = 5000 instructions, 2 stalls

Loop unrolling

1000

```
for (i=999; i>=0; i--)
   x[i] = x[i] + y;
```

← loop

i = 1

Unroll loop once

1, 0

x[0] ==

```
for (i=999; i>=0; i=i-2)
{
   x[i] = x[i] + y;
   x[i-1] = x[i-1] + y;
}
```

} Compute.

Half the iterations

Once

Loop unrolling

LD F0, 0(R1
LD F0, 8(R1)
Add F4, F0, F2
Add F4, F0, F2

```
for (i=999; i>=0; i=i-2)
{
  x[i] = x[i] + y;
  x[i-1] = x[i-1] + y;
}
```

```
Loop:
L.D  F0, 0(R1)
ADD.D     F4, F0, F2
S.D  F4, 0(R1)
L.D  F0, 0(R1)
ADD.D     F4, F0, F2
S.D  F4, 0(R1)
SUBI   R1, R1, #8
BNEQ  R1,R2 Loop
```

LD F0, 0(R1)

→ 1 ST

→ 3 ST.

8

→ 1 ST

→ 3 ST

??

Control hazards.

Unroll once

Unroll loop.

↓ Reg. Rename

↓ Reorder instr.

```
for (i=999; i>=0; i=i-2)
{
   x[i] = x[i] + y;
   x[i-1] = x[i-1] + y;
}
```

```
Loop:
L.D    F0, 0(R1)
ADD.D      F4, F0, F2
S.D    F4, 0(R1)
L.D    F6, -8(R1)
ADD.D      F8, F6, F2
S.D    F8, -8(R1)
SUBI   R1, R1, 16
BNEQ   R1,R2 Loop
```

Increase in instructions ?

Unroll once --> half the iterations

```
for (i=999; i>=0; i=i-2)
{
   x[i] = x[i] + y;
   x[i-1] = x[i-1] + y;
}
```

x(i-2) = x(i-2)+y

LD
add
SD

```
Loop:
L.D     F0, 0(R1)
ADD.D       F4, F0, F2
S.D     F4, 0(R1)
L.D     F6, -8(R1)
ADD.D       F8, F6, F2
S.D     F8, -8(R1)
SUBI    R1, R1, 16
BNEQ    R1,R2 Loop
```

LD
Add
SD

8 * 500 = 4000 instructions
Stalls?

stalls

```
L.D     F0, 0(R1)
Stall
ADD.D     F4, F0, F2
3 stalls
S.D     F4, 0(R1)
L.D     F6, -8(R1)
Stall
ADD.D     F8, F6, F2
3 stalls
S.D     F8, -8(R1)
SUBI    R1, R1, 16
Stall
BNEQ    R1,R2 Loop
```

Re-ordering/Scheduling

↑ ↓ stalls.
↑ ↓ Control hazards
Codesize
Reg pressure

```
L.D      F0, 0(R1)
Stall
ADD.D    F4, F0, F2
3 stalls
S.D      F4, 0(R1)
L.D      F6, -8(R1)
Stall
ADD.D    F8, F6, F2
3 stalls
S.D      F8, -8(R1)
SUBI     R1, R1, 16
Stall
BNEQ     R1,R2 Loop
```

(4)-Add.

I-cache.

→

```
L.D      F0, 0(R1)
L.D      F6, -8(R1)      -->
renaming useful here
ADD.D    F4, F0, F2
ADD.D    F8, F6, F2
SUBI     R1, R1, 16
stall
S.D      F4, 16(R1)
S.D      F8, 8(R1)
BNEQ     R1,R2 Loop
```

(4)

Add

SD

1024x1024

1024 clk
16

Reduced stalls--> better CPI
Reduced instructions 8 * 500 = 4000
Exec time = Instr * CPI * clk cycle time

Unroll ~~thrice~~ twice

Need to change reg names --> 16 stalls

```
L.D     F0, 0(R1)
ADD.D   F4, F0, F2
S.D     F4, 0(R1)
L.D     F6, -8(R1)
ADD.D   F8, F6, F2
S.D     F8, -8(R1)
L.D     F0, 0(R1)
ADD.D   F4, F0, F2
S.D     F4, 0(R1)
L.D     F0, 0(R1)
ADD.D   F4, F0, F2
S.D     F4, 0(R1)
SUBI    R1, R1, 16
BNEQ    R1,R2 Loop
```

Schedule/Re-order
No stalls

Twice

Unrolled code

L.D F0, 0(R1)
L.D F6, -8(R1)
Load
~~Load~~
ADD.D F4, F0, F2
1 ADD.D F8, F6, F2
2 Add
~~Add~~
3 SUBI R1, R1, 16
S.D F4, 16(R1)
S.D F8, 8(R1)
Store
~~Store~~
BNEQ R1,R2 Loop

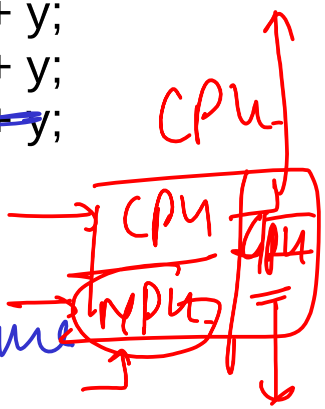
```
for (i=999; i>=0; i=i-2)
{
   x[i] = x[i] + y;
   x[i-1] = x[i-1] + y;
   x[i-2] = x[i-2] + y;
   ~~x[i-3] = x[i-3] + y;~~
}
```

MACmul Rename

MAC

SPMU

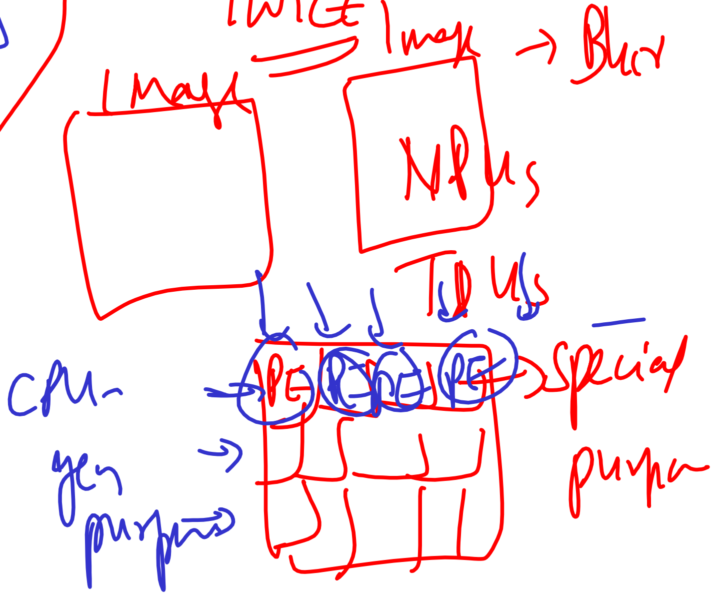
Reorder

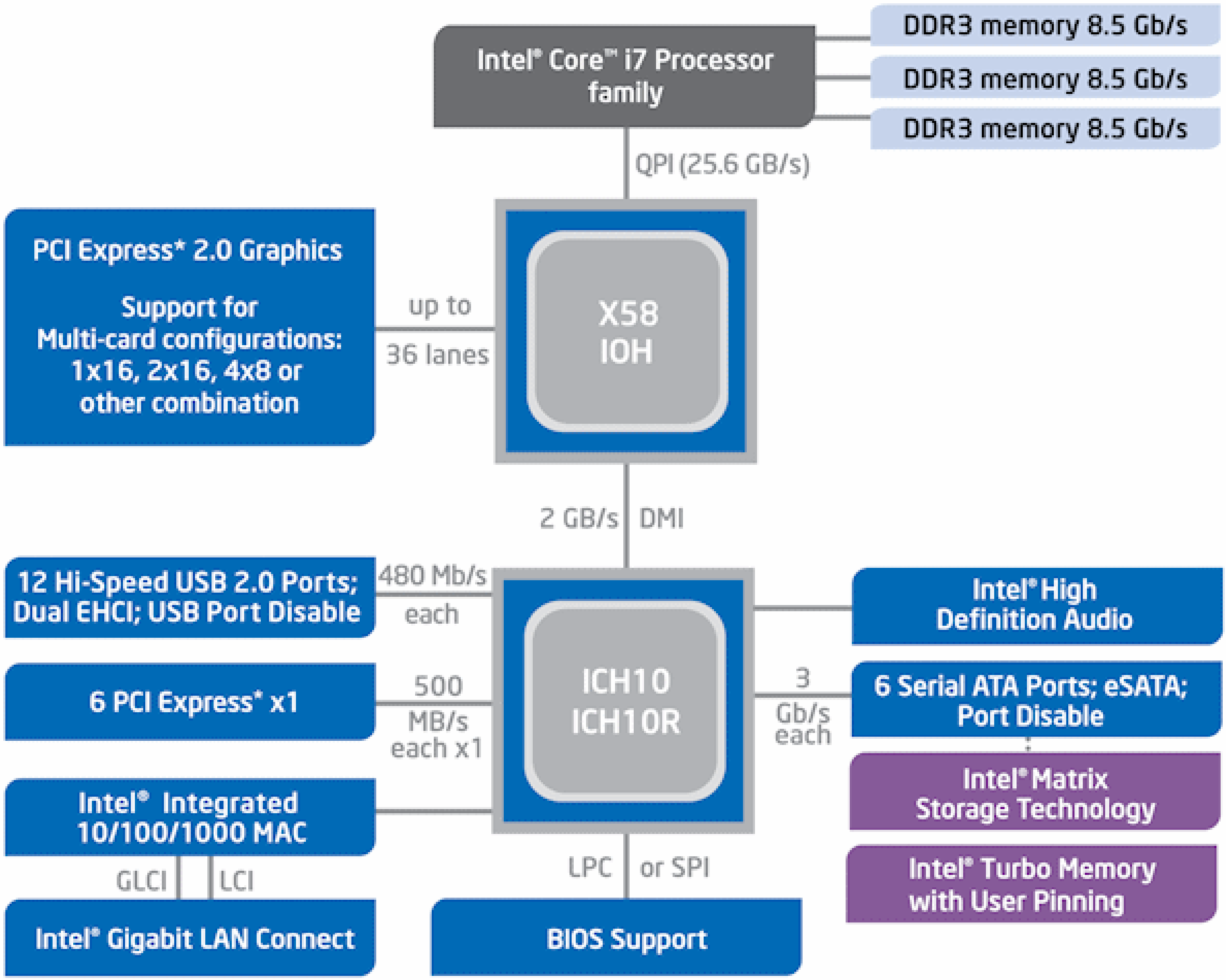


abc

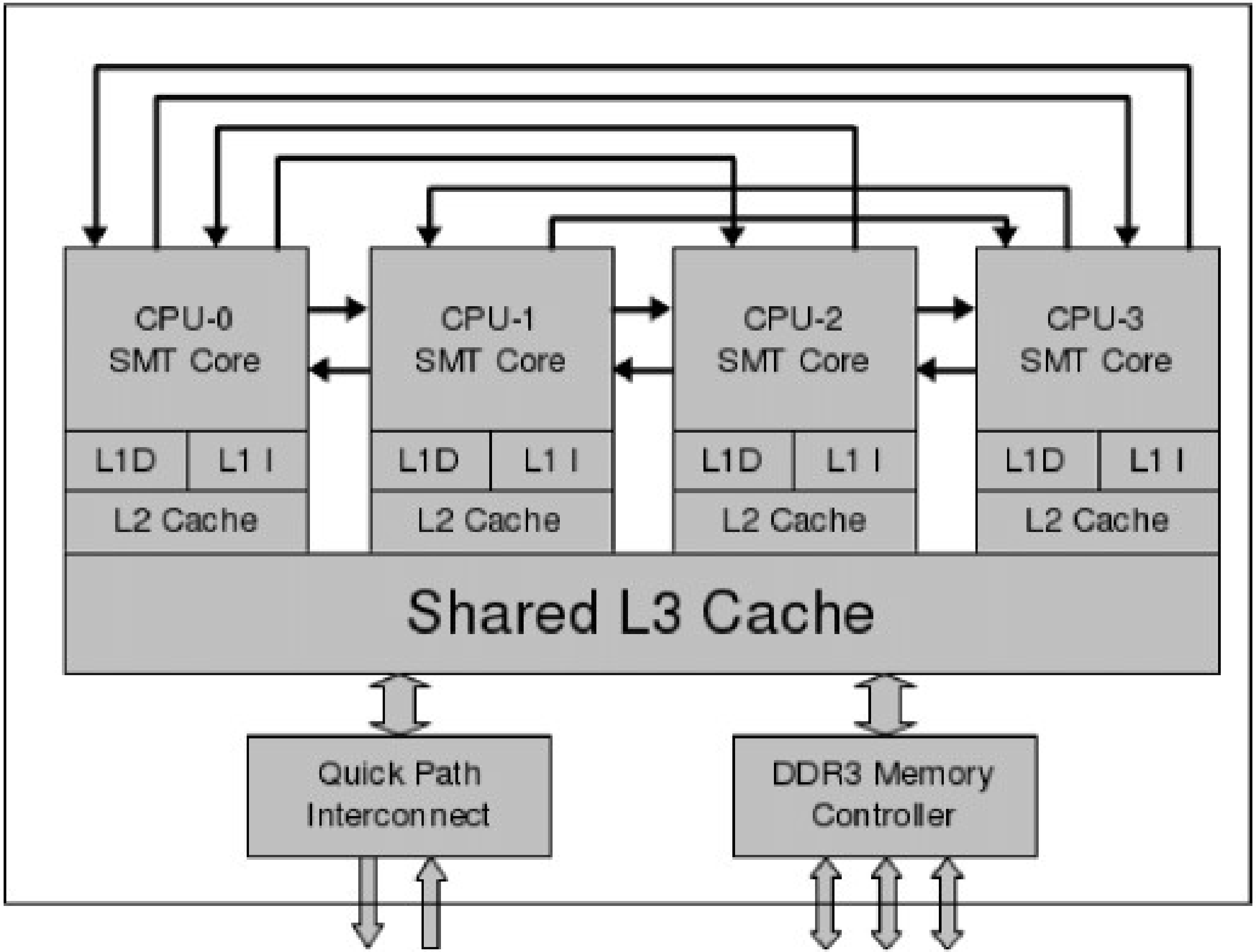
EPYC

TWICE



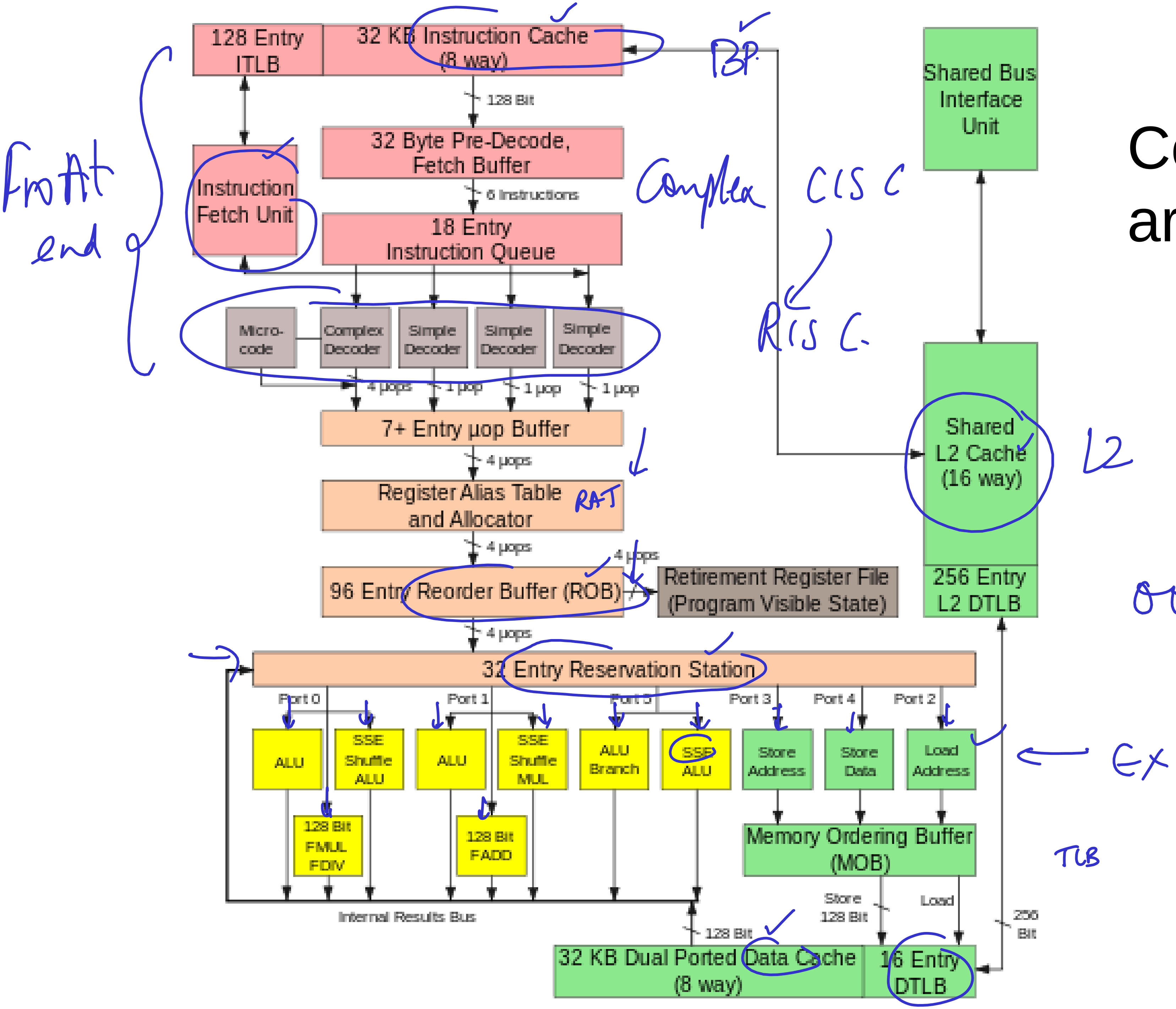

Intel® Core™ i7 Processor family
DDR3 memory 8.5 Gb/s
DDR3 memory 8.5 Gb/s
DDR3 memory 8.5 Gb/s
QPI (25.6 GB/s)
PCI Express* 2.0 Graphics
Support for Multi-card configurations: 1x16, 2x16, 4x8 or other combination
up to 36 lanes
X58 IOH
2 GB/s DMI
12 Hi-Speed USB 2.0 Ports; Dual EHCI; USB Port Disable
480 Mb/s each
6 PCI Express* x1
500 MB/s each x1
ICH10 ICH10R
Intel® High Definition Audio
3 Gb/s each
6 Serial ATA Ports; eSATA; Port Disable
Intel® Matrix Storage Technology
Intel® Turbo Memory with User Pinning
Intel® Integrated 10/100/1000 MAC
GLCI
LCI
Intel® Gigabit LAN Connect
LPC or SPI
BIOS Support

http://onlyusama.blogspot.com/2009/06/block-diagram-of-core-i7-system.html



Evaluating Multicore Architectures for Application in High Assurance Systems - ResearchGate

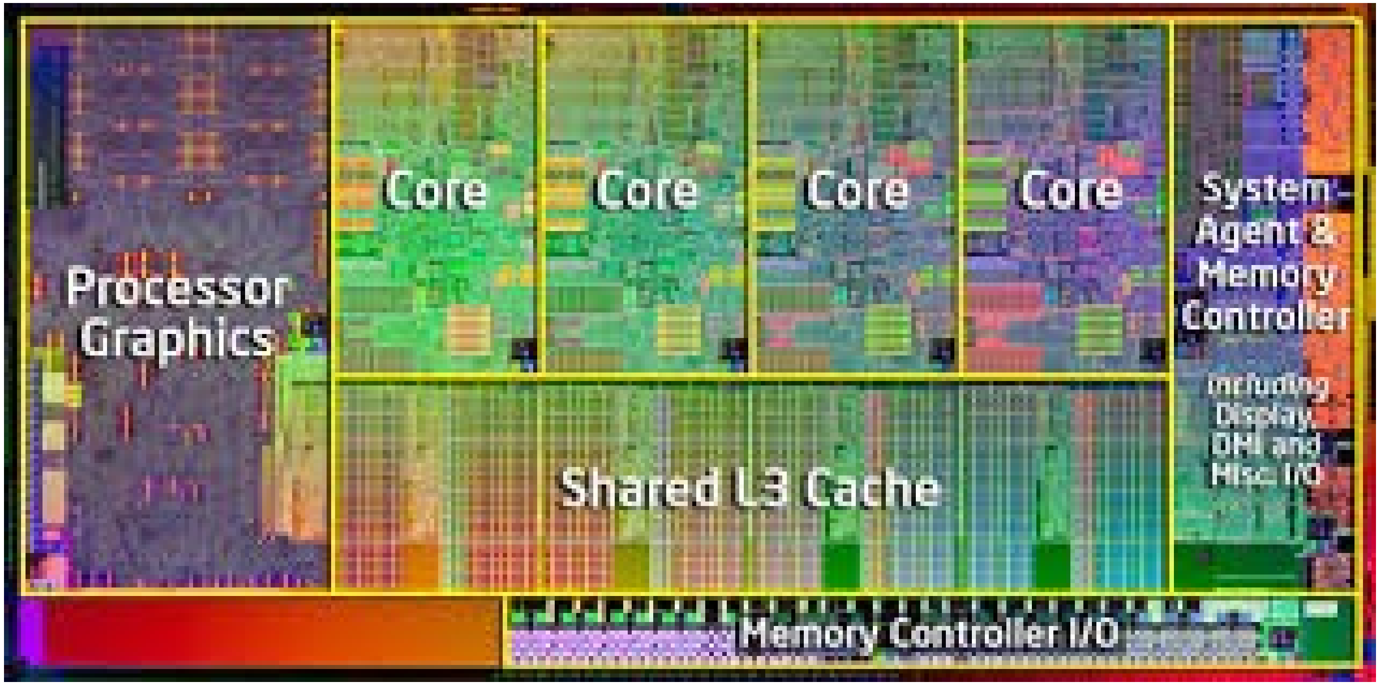
Core2 architecture

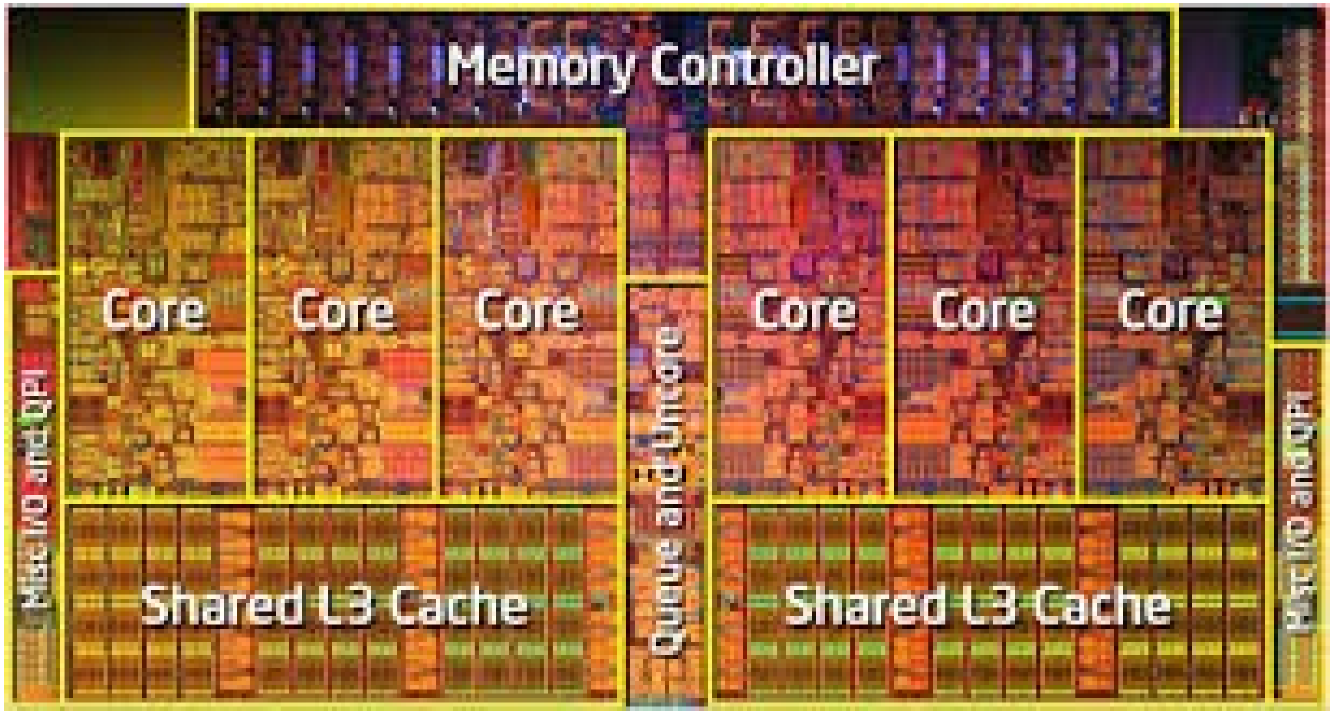


Intel Core 2 Architecture

OoO processor

Chapter 4 — The Processor H&P Wiki





https://www.cs.uaf.edu/2009/fall/cs441/proj1/russell/index.html

Acknowledgements

- Compiler Optimizations for Exposing ILP – IIT Madras
- Loop Unrolling Benefits - Georgia Tech