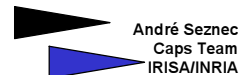




# High Performance Microprocessors

André Seznec  
IRISA/INRIA  
<http://www.irisa.fr/caps>

1



High Performance Microprocessors

## Where are we (Fall 2000)?

- 400 Mhz to 1.1 Ghz
- 1 cycle = crossing the ALU + bypassing
- a floating point operation = 2-4 cycles
- register file read/write : 1 cycle
  - 2 -3 in next generation
- L1 cache read/write: 1-3 cycles
  - trade-off size-associativity-hit time

2



## Where are we (Fall 2000) ?

- Integration : 0.25 $\mu$ , 0.18 $\mu$ , 0.125 $\mu$  (2001)
- 10 -20 millions of transistors in logic
- Memory structures: up to 150 millions of transistors
- 20 to 60 Watts
  - > 100 W soon
- 400-600 pins
  - > 1000 soon

3

## Where are we (Fall 2000) ?

- Processors x86 :
  - low end : 500 Mhz, <100 \$
  - high end : 1 Ghz, 1000 \$
- DRAM : 1\$ per Mbyte
- SRAM : 50\$ per Mbyte

4

## Binary compatibility

- 300 000 000 PCs !
  - Introducing a new ISA: a risky business !!
- It might change :
  - embedded applications (cell phones, automotive, ..)
  - 64 bits architectures are needed

5

## 32 or 64 bits architecture

- Virtual addresses computed on 32 bits
  - PowerPC, x86,
- Virtual addresses computed on 64 bits
  - MIPS III, Alpha, Ultrasparc, HP-PA 2.x, IA64
  - In 2005: Games ? Word ?
- Can 't be avoided :
  - x86 ?

6

## Which instruction set ?

- CISC: x86
- RISC: Sparc, Mips, Alpha, PowerPC, HP-PA
- EPIC: IA-64
  
- does it matter ?

7

## RISC ISA

- a single instruction width : 32 bits
  - simple decode
  - next address
- Load/store architecture
- Simple addressing : based and indexed
- register-register
- Advantage: very simple to pipeline

8

## RISC (2)

- general registers + floating point registers
- memory access: 1-8 bytes ALIGNED
- limited support to speculative execution

9

## CISC x86

- variable instruction size
- operands in registers and in memory
- destructive code: write one of its operand
- non-deterministic instruction execution time

10

## EPIC IA64

- EPIC IA64 =
  - RISC
  - + lot of registers (128 INT , 128 FP)
  - + 3 instructions encoded on 128 bits bundle
  - + 64 predicate registers
  - + hardware interlocks
  - + (?) Advanced Loads
- The compiler is in charge of the ILP

11

## EPIC IA 64

- Not so good :
  - - ISA support for software pipeline
    - Rotating registers
    - Loop management
  - -Register windows: with variable size !
  - -No based addressing
  - - post-incremented addressing
  - - (?) Advanced Loads

12

## ISA: does it matter?

- 32 or 64 bits:
  - towards x86 death (or mutation !)
- Performances:
  - x86 ? :=)
  - floating point !!
  - At equal technology ?

13

## ISA: does it matter (2) ?

- x86: translation in  $\mu\text{Op}$ 
  - 4 lost cycles !
  - Or use a trace cache
- x86: too few floating point registers
- Alpha 21264: 2 operands 1 result
  - + performant RISC
- Itanium: IA 64 in order
  - 800 Mhz in  $0.18 \mu$
  - Alpha 21164 700 Mhz  $0.35 \mu$  (1997)

14

## So ?

- x86: + installed base, + Intel, - must migrate to 64 bits
- IA64: + speculative execution, + Intel
- Alpha: + « clean » ISA, - what is Compaq strategy ?
- Sparc: = ISA, + SUN
- Power(PC): complex ISA, =IBM and Motorola

15

## The architect challenge

- 400 mm<sup>2</sup> of silicon
- 3 technology generations ahead
- What will you use for performance ?
  - Pipeline
  - ILP
  - Speculative execution
  - Memory hierarchy
  - Thread parallelism

16

## Deeper and deeper pipeline

- Less gates crossing in a single cycle.
- 1 cycle = ALU crossing + feeding back
- Longer and longer intra-CPU communications :
  - several cycles to cross the chip
- More and more complex control:
  - more instructions in //
  - more speculation

17

## A few pipeline depths

- 12-14 cycles on Intel Pentium III
- 10-12 cycles on AMD Athlon
- 7-9 cycles on Alpha 21264
- 9 cycles on UltraSparc
  
- 10 cycles on Itanium
- 20 cycles on Willamette
  
- And it is likely to continue !!

18

## Instruction Level Parallelism

- Superscalar:
  - the hardware is responsible for the parallel issuing
- binary compatibility
- All general purpose processors are superscalar

19

## The superscalar degree

- Hard to define
  - « performance that can not be exceeded »
- 4 inst / cycles on almost all existing processors
- 6 inst / cycles on Itanium
- 8 inst / cycles on (future) Alpha 21464
- 16 -32 on Alpha 21964 ?? En 2012 ? :=)

20

## Superscalar: the difficulties

- ILP is limited on applications:: 3 to 8 instructions per cycle
- Register files :
  - number of ports is increasing
  - access is becoming a critical path
- How to feed FUs with instructions ? with operands?
- How to manage data dependencies
- branch issues

21

## In order or out of order execution

- In order execution :
  - Ah! If all latencies were statically known ..
- In order execution:
  - UltraSparc 3, Itanium
  - The compiler must do the work
- Out-of-order execution
  - Alpha 21264, Pentium III, Athlon, Power (PC), HP-PA 8500

22

## Why in order execution

- simple to implement
  - less transistors
  - shorter design cycle
  - faster clock (*maybe*)
- shorter pipeline
- the compiler « sees » everything:
  - micro-architecture
  - application

23

## Why out-of-order execution

- Static ILP is limited on non-regular codes
- The compiler does not see everything :
  - unknown latencies at compile time
  - memory (in)dependencies unknown at compile time
- No need for recompiling
  - hum !!

24

## Out-of-order execution (1)

- Principle :
  - execute ASAP: available operands and FUs
- Implementation :
  - large instruction window for selecting fireable instructions

25

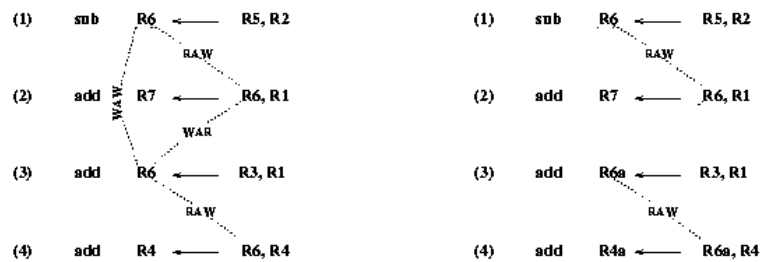
## Out-of-order execution (2)

- Sequencing consists of:
  - // read of instructions
  - mark dependencies
  - rename registers
  - dispatch to FUS
  - waiting ..
- Dependency managing use space and time

26

## Renaming registers: removing false dependencies

- WAW and WAR register hazards are not true dependencies.



27

## Memory dependencies

- To execute a store, data to be stored is needed
- Every subsequent load is potentially dependent on any previous store
- Solution (old):
  - addresses are computed in program order
  - Loads bypass stores with hazards detection

28

## Memory dependencies (2)

- Solution (current):
  - optimistic execution (out-of-order)
  - Repaired when true dependency is encountered
  - Pb: repair cost
- Next generation: dependencies are predicted
  - Moshovos et Sohi, Micro'30, 1997
  - Chrysos et Emer, ISCA '26, 1998

29

## Control hazards

- 15 -30% of instructions are branches
- Target and direction are known very late in the pipeline :
  - Cycle 7 on DEC 21264
  - Cycle 11 on Intel Pentium II
  - Cycle 18 on Willamette
- Don 't waste (all) these cycles !

30

## Dynamic branch prediction

- Use history to predict the branch and fetch subsequent instructions
- Implementation: tables read in // with the I-Cache
- Will be predicted:
  - conditional branch target and direction
  - procedure returns
  - indirect branches

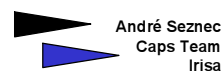
31



## Dynamic branch prediction

- 1992: DEC 21064, 1 bit scheme
- 1993: Pentium, 2 bits scheme
- 1995: PentiumPro, local history
- 1998: DEC 21264, hybrid predictor

32



## Dynamic branch prediction : general trend

- More and more complex schemes
- target prediction and direction prediction are decoupled
- Return stack for procedures
- ISA support for speculative execution
  - CMOV
  - IA64 predicates

33

## Out-of-order execution: « undoing »

- uncorrect branch prediction
- uncorrect independency prediction
- Interruption, exception
  
- Validate in program order
- No definitive action out-of-order

34

## Being able to « undo »

- A copy of the register file is updated in program order
- or
- a map of logical registers on physical registers is saved
- Stores are done at validation

35

## Memory hierarchy may dictate performance

- Example :
  - 4 instructions/cycle,
  - 1 memory access per cycle
  - 10 cycles miss penalty if hit on the L2 cache
  - 50 cycles for accessing the memory
  - 2% instruction L1 misses , 4% data L1 misses, 1/4 misses on L2
- 400 instructions : 395 cycles

36

## Hum ..

- L1 caches are non-blocking
  - L2 cache is pipelined
  - memory is pipelined
  - hardware and software prefetch
- 
- Latency and throughput are BOTH important

37

## L1 caches: general trend

- 1-2 cycles for read or write
  - multiple ported
  - non blocking
  - small associativity degree
- 
- will remain small

38

## L2 caches: general trend

- mandatory !
- on-chip or on the MCM
- Pipelined access
  - « short » latency: 7-12 cycles
  - 16 bytes bus WILL GROW
  - cycle time: 1-3 processor cycles
- contention on the L2 cache might become a bottleneck

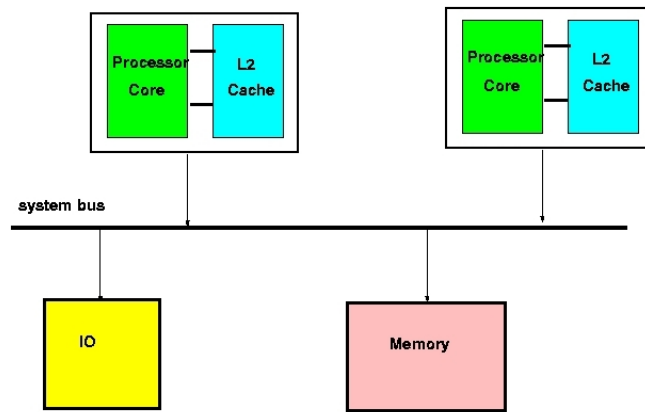
39

## Main memory

- Far, very far, too far from the core:
  - several hundred instructions
- Memory on the system bus: too slow !!
- Memory bus + system bus

40

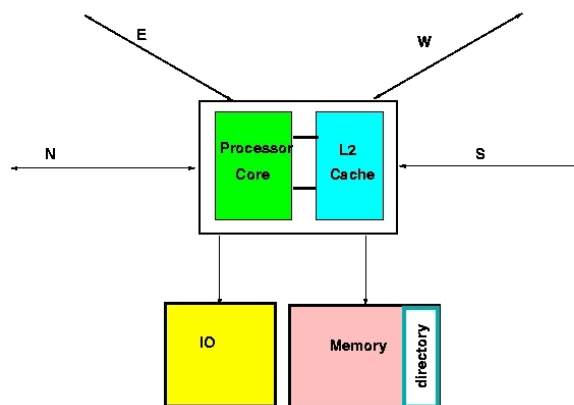
## Main memory on the system bus



41

André Seznec  
Caps Team  
Irisa

## Glueless memory connection



42

André Seznec  
Caps Team  
Irisa

## How to spend a billion transistors?

- IRAM: a processor and its memory
- Ultimate speculation on a uniprocessor
- Thread parallelism:
  - On-chip multiprocessor
  - SMT processor

43

## IRAM

- Processor and memory on a single die
  - huge memory bandwidth
- Not for general purpose computing
  - Memory requirements increase
  - How it scales ?
- Technological issue: mixing DRAM and logic
- Cost-effective solution for lot of embedded applications

44

## Ultimate speculation on a uniprocessor

- 16 or 32 way superscalar processor
- hyperspeculation:
  - branches, , dependencies, data ...
- Challenges::
  - prediction accuracy
  - communications on the chip
  - contention :
    - caches , registers, ...

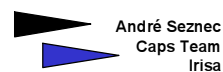
45



## Thread parallelism on the chip: it 's time now !

- On-chip thread parallelism is possible and will arrive
- on-chip multiprocessor ?
  - IBM Power 4 ( fin 2001)
- Simultaneous Multithreading ?
  - Compaq Alpha 21464 ( 2003)

46



## On-chip multiprocessor

- May be the solution, but..
  - Memory bandwidth?
  - Single thread performance ?

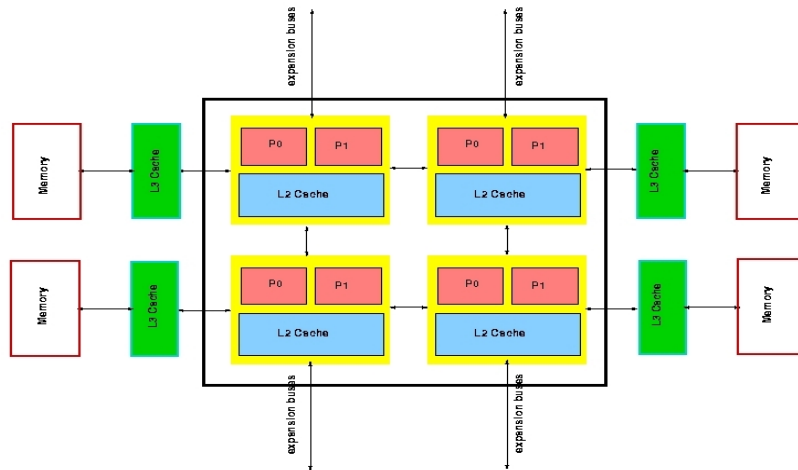
47

## On-chip multiprocessor: IBM Power 4 (2001)

- Market segment: servers
- 2 superscalar processors on a chip:
  - shared L2 cache
- 4 chips on a MCM (multichip module)
  - huge bandwidth on the chip and on the MCM

48

## Programmer view



49

## Simultaneous Multithreading (SMT)

- FUs are underused
- SMT:
  - Share FUs from a superscalar processor among several threads
- Advantages:
  - 1 thread may use all the resources
  - structures are dynamically shared (caches, predictors, FUs)

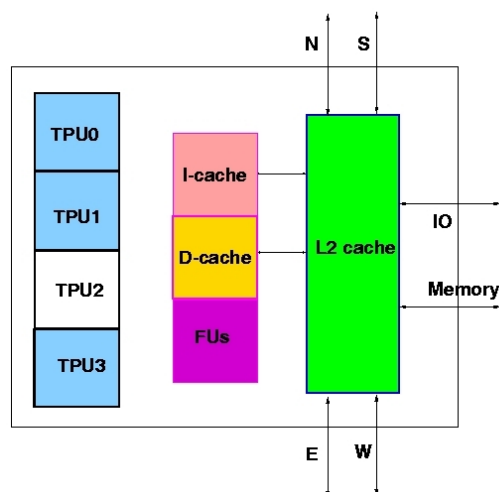
50

## SMT: Alpha 21464 (2003)

- 8 way superscalar
  - Ultimate performance on a thread
- if multithread or multiprogrammed, 4 threads share the hardware:
  - overhead 5-10 %

51

## Programmer view



52

## SMT: research directions

- Speculative multithreading:
  - one step farther than branch prediction
- Multipath execution:
  - let us execute the two pathes of a branch
- Subordinate thread
  - a slave thread works for the principal thread:
    - prefetching
    - branch anticipation

53

## Predicting the future

### PC processor in 2010

- x86+ ISA
  - mode 64 bits
- 10 way superscalar SMT
- < 50 mm<sup>2</sup>
- cache snooping

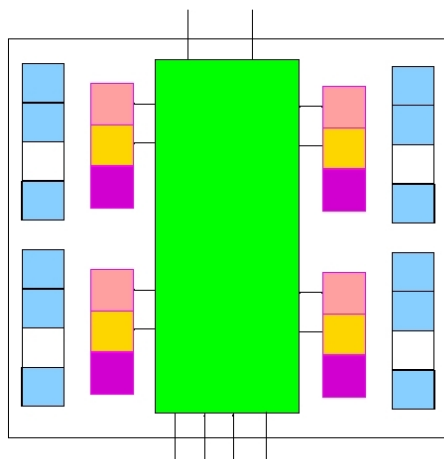
54

## Predicting the future The server processor in 2010

- IA64 or Alpha+ ISA
  - predicate registers
- Multi 10 way SMT
- directory coherence

55

## Programmer view !



56

## Challenges

- Design complexity?
- How to program (compile) for these monsters?