ILP Ends TLP Begins

Today's topics:

Explore a perfect machine

unlimited budget to see where ILP goes

answer: not far enough

Look to TLP & multi-threading for help

everything has it's issues

we'll look at some of them

Apology

a bit more data than usual

try not to yawn LOUDLY

ILP Limits via an Oracle

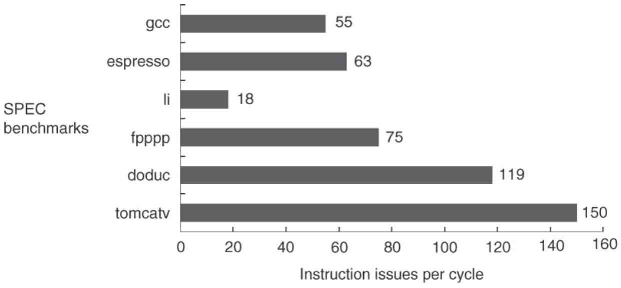
- Suspend reality and think of a perfect machine
  - infinite number of rename registers
    - » no Wax hazards
    - » for window size of n: n^2-n comparisons for each register field
  - perfect branch & jump prediction
    - » unbounded buffer of instructions available for execution
  - perfect address alias analysis
    - » independent loads can be moved ahead of stores
  - perfect L1$'s
    - » hit in 1 cycle
  - as many XU's as will ever be needed
    - » no structural stalls
- Infinite cost unrealistic
  - simulate rather than build
    - » allows exploration
      - just how far can we get with ILP on a perfect machine
      - and sequential code

IBM Power5

- Most advanced superscalar processor to date
  - 4 fetch
  - 6 issue
  - 88 integer rename regs, 88 float rename regs
  - pipeline has over 200 instructions in flight
    - » including 32 loads and 32 stores
- Not quite the Oracle but on the way
  - consumes a lot of power
  - target is blade server segment

ILP w/ Infinite Window Size

- Looks great
  - when compared w/ today's IPC < 3
    - » if you can ignore the infinite cost



© 2007 Elsevier, Inc. All rights reserved.

Limit Window Size

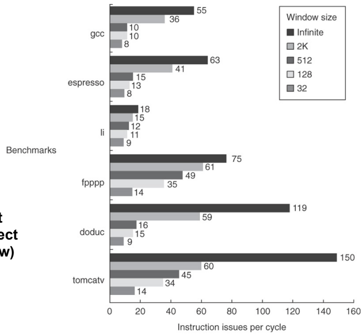
- **ILP shrinks rapidly**

2K = ~ 4M * 3 5-bit compares
512 = 785K compares
128 = 49K compares
32 = 3K compares

Plus compares happen every cycle

conclude 32 is doable but watty

can improve by maybe 3x at great cost (remember this is still a perfect machine – just w/ a limited window)



© 2007 Elsevier, Inc. All rights reserved.

Switch to Half-Infinity

- **You do the math**
- **For the remaining data assume**
  - **2K window size**
    - » **~12 M 5-bit compares every clock**
    - » **>10x bigger than anything that's been built**
  - **64 issue**
    - » **~10x more than anything real**
- **Why choose this**
  - **given other restrictions it won't be a limit**
    - » **can you say "easier to simulate"**
    - » **I knew you could**

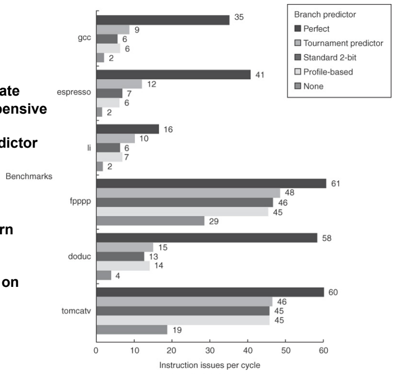
Look at Semi-Real Branch Prediction

Tnmt: 8K entry predictor
Jump predictor: 2K entries
48K bits and 3% mispredict rate which is very good – just expensive

Standard: 512 entry 2-bit predictor

Conclusion:
have to predict integer codes are a problem yet highly important in modern data-center apps

Nobody makes much money on floating point – sad reality



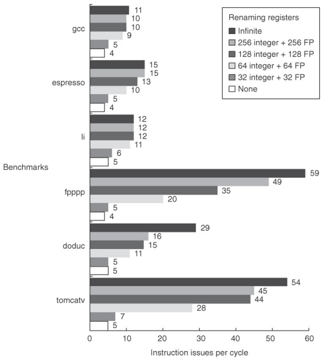
© 2007 Elsevier, Inc. All rights reserved.

Limiting Rename Registers

Integer codes remain problematic

$ problem with FP remains but ILP looks good if you don't care about $

conclusion – need around 64 renamed registers to make much of a difference

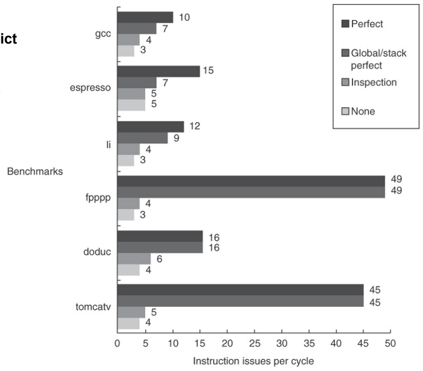


© 2007 Elsevier, Inc. All rights reserved.

Alias Analysis Influence

GL/STK – heap ref's conflict but nothing else

Inspection – what can the compiler do?



© 2007 Elsevier, Inc. All rights reserved.

Ambitious but Possible?

- HAL
  - 1 better than IBM in all letters
  - 64 issue no restrictions
    - » this one is actually ridiculous
      - it does focus on ILP limits rather than structural stalls
  - 1K entry tournament predictor
    - » this has been done
  - perfect disambiguation
    - » note close to possible for small window sizes
    - » impractical for large windows
  - 64 register rename pool
    - » ~100 have been done
- What do we get?
  - Al the Harpy says "a really good heater"

And VIOLA!

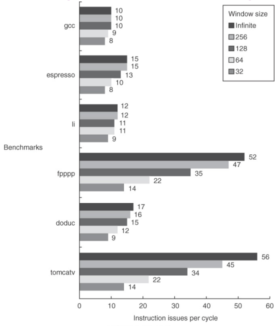
- 3-4x improvement
  - for a machine nobody will buy – too hot, too costly

Interesting study & clear conclusion:

ILP is already past the point of diminishing return

Programmer is going to need to help out w/ exposing parallelism

Need a different type of HW support for parallelism



© 2007 Elsevier, Inc. All rights reserved.

Enter TLP

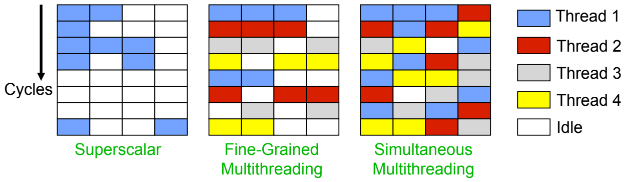
- Again not a new idea
  - been around for > 10 years
    - » Tullsen – UW – 1995 publishes the SMT idea
    - » TERA MTA & IBM Pulsar show up in late 90's – both MT
- Thread vs. Process confusion
  - process runs in it's own virtual memory space
    - » no shared memory
    - » lots of OS protection & overhead
    - » communicate via "message like channels" – e.g. pipes in Unix
  - threads
    - » share memory and therefore synchronization needed
  - both are independent entities
    - » with their own sets of registers and process state
  - TLP difference
    - » multiple threads can run concurrently or interleaved on the same processor
    - » one at a time and context switch for processes

Multi-Threading

- **2 variants**
  - **fine-grained MT – e.g. TERA**
    - » **round robin walk through threads**
      - **next cycle – next thread**
    - » **TERA – 128 threads**
      - **built in 128 cycle load-use delay**
        - – **basic idea was to cover main memory latency and do away w/ caches**
        - – **great if you put every app into a 128 thread mold**
      - **it failed and Burton goes to the dark side (a.k.a. Microsoft)**
  - **coarse grained MT – e.g. IBM Pulsar**
    - » **sometimes called "switch on miss"**
    - » **basic idea**
      - **anytime something bad happens**
        - – **TLB or L2 miss**
      - **switch to next runable thread**
        - – **some sort of fairness policy is required**
        - – **usually just round-robin suffices**
      - **similar goal – hide performance effect of long stalls**

Symmetric Multithreading

- **Idea**
  - **multiple independent threads**
  - **increase number of parallel instructions to issue**

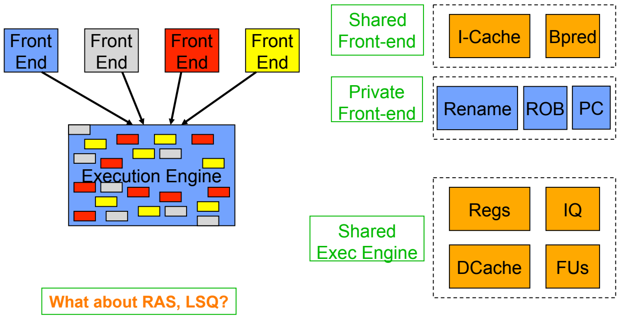


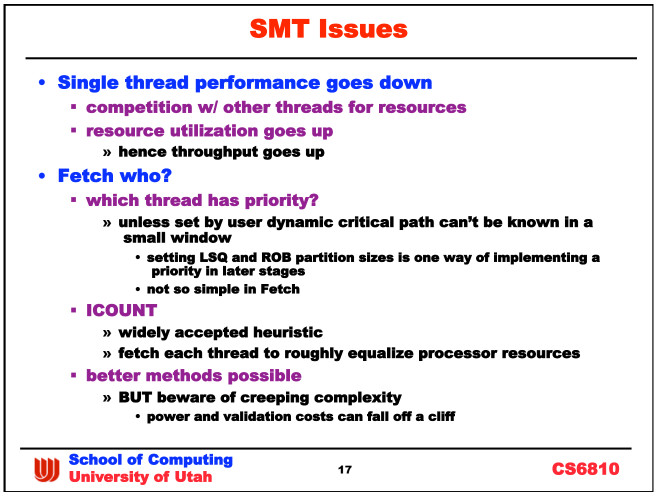
- » **superscalar – not enough ILP and idle on cache miss**
- » **FGMT – not enough ILP in any one thread**
- » **SMT – improves since broader set of independent instructions**
  - **programmer supplied parallelism**
  - **takes advantage of dynamic issue superscalar tactics**

SMT Resource Perspective

- **Each thread has it's own**
  - **PC, next PC**
    - » **next is needed for exceptions**
  - **private logical registers**
    - » **and mapping to renamed physical registers**
  - **ROB**
    - » **if shared a stall in one thread will stall the others**
- **Shared**
  - **branch predictor**
    - » **larger size will be needed**
  - **main memory ports, TLB, page table**
    - » **artifact of shared memory**
    - » **more threads does increase memory pressure**
      - **biggest problem is single ported L1$'s**

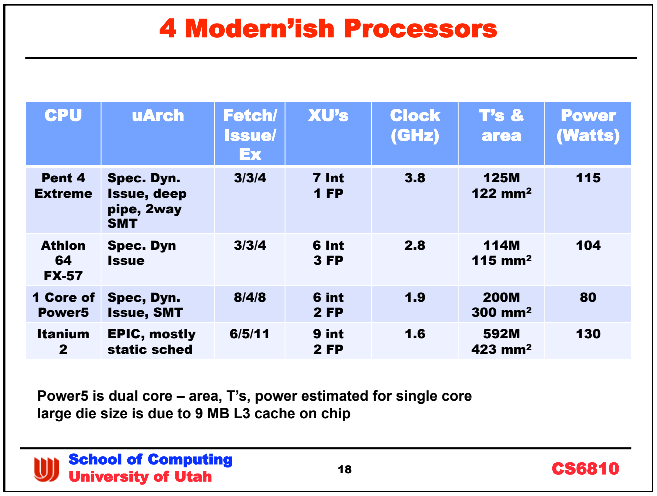
SMT Pipeline Structure





SMT Issues

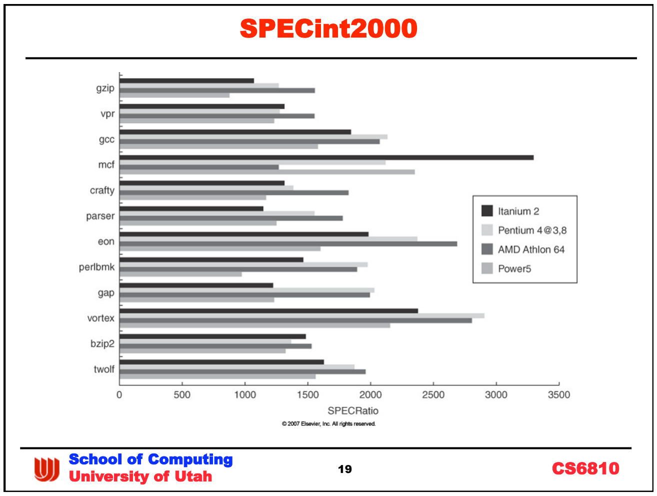
- **Single thread performance goes down**
  - **competition w/ other threads for resources**
  - **resource utilization goes up**
    - » **hence throughput goes up**
- **Fetch who?**
  - **which thread has priority?**
    - » **unless set by user dynamic critical path can't be known in a small window**
      - **setting LSQ and ROB partition sizes is one way of implementing a priority in later stages**
      - **not so simple in Fetch**
  - **ICOUNT**
    - » **widely accepted heuristic**
    - » **fetch each thread to roughly equalize processor resources**
  - **better methods possible**
    - » **BUT beware of creeping complexity**
      - **power and validation costs can fall off a cliff**



4 Modern'ish Processors

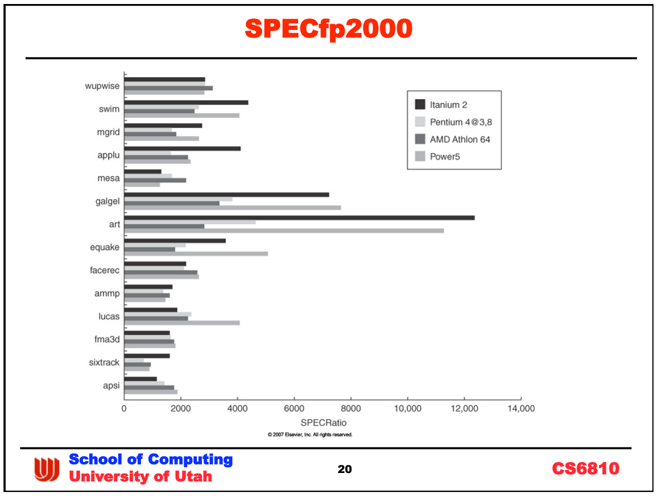
CPU	uArch	Fetch/ Issue/ Ex	XU's	Clock (GHz)	T's & area	Power (Watts)
Pent 4 Extreme	Spec. Dyn. Issue, deep pipe, 2way SMT	3/3/4	7 Int 1 FP	3.8	125M 122 mm^2	115
Athlon 64 FX-57	Spec. Dyn Issue	3/3/4	6 Int 3 FP	2.8	114M 115 mm^2	104
1 Core of Power5	Spec, Dyn. Issue, SMT	8/4/8	6 int 2 FP	1.9	200M 300 mm^2	80
Itanium 2	EPIC, mostly static sched	6/5/11	9 int 2 FP	1.6	592M 423 mm^2	130

Power5 is dual core – area, T's, power estimated for single core
large die size is due to 9 MB L3 cache on chip



SPECint2000

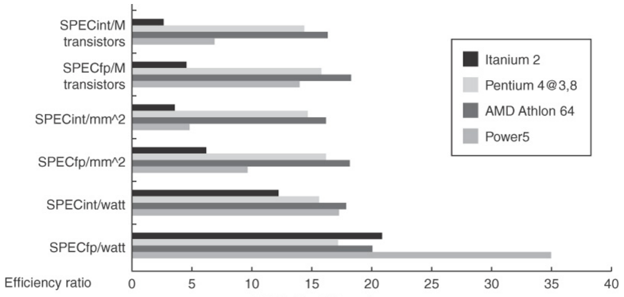
© 2007 Elsevier, Inc. All rights reserved.



SPECfp2000

© 2007 Elsevier, Inc. All rights reserved.

Efficiency



© 2007 Elsevier, Inc. All rights reserved.

Note: as we move into multi-core perf/watt becomes the critical efficiency metric
Even better: energy-delay product since that is architecture and workload specific

Concluding Remarks

- SMT is a big boost to the ILP game
  - uses previous skills in dynamic superscalar architecture
  - rules of thumb
    - » double threads
      - 1.6x performance gain
      - ~10-15% power gain
  - where does the curve saturate
    - » depends on workload
      - ~4/core seems to be a sweet spot
      - time will tell
    - » Sun Niagra Falls
      - 8 cores, 8 threads/core
        - – simpler cores however
      - performs well in the data-center
- Next
  - leave processor side and examine the memory side
    - » both need to be balanced and done right to win