CS184b: Computer Architecture (Abstractions and Optimizations)

Day 5: April 14, 2003
ILP 2

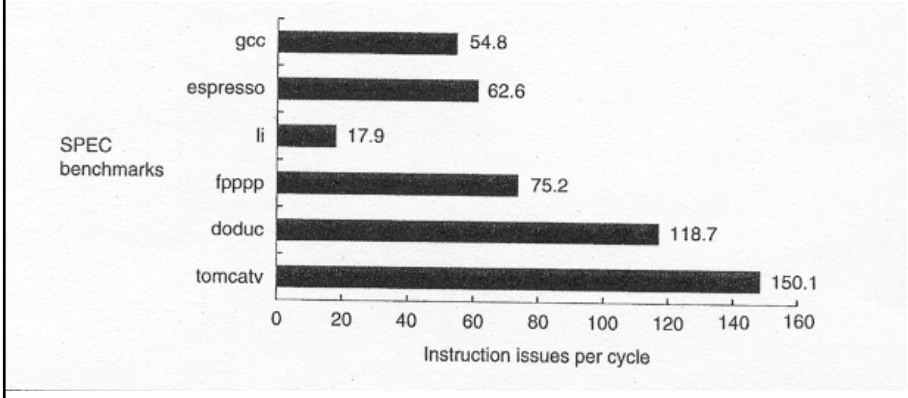
Today

- ILP Limits
- Practical Issues
  - Finite size issues
- Cost Scaling
- Ultrascalar

Limit Studies

- **Goal:** understand how far you can go
  - this case, how much ILP can find
- Remove current/artificial limits
  - do full renaming, arbitrary look ahead
  - perfect control prediction, memory disambiguation
- Careful with assumptions
  - can still be pessimistic
  - is there another way to do it?
  - Another way around the limitation?

Available ILP



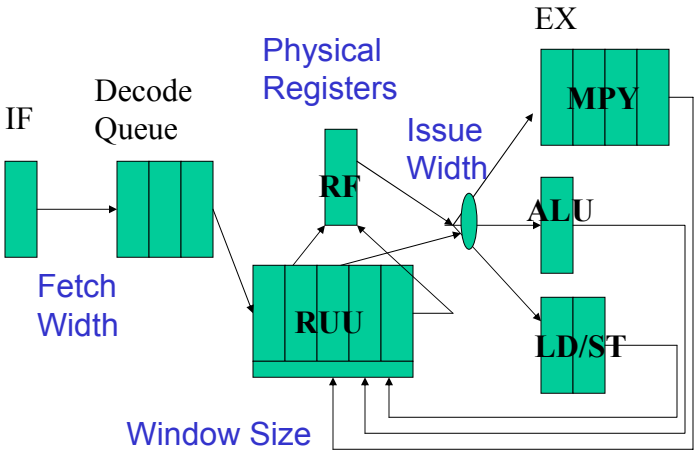
[Hennessy and Patterson 4.38e2/3.35e3]

What do we achieve today?

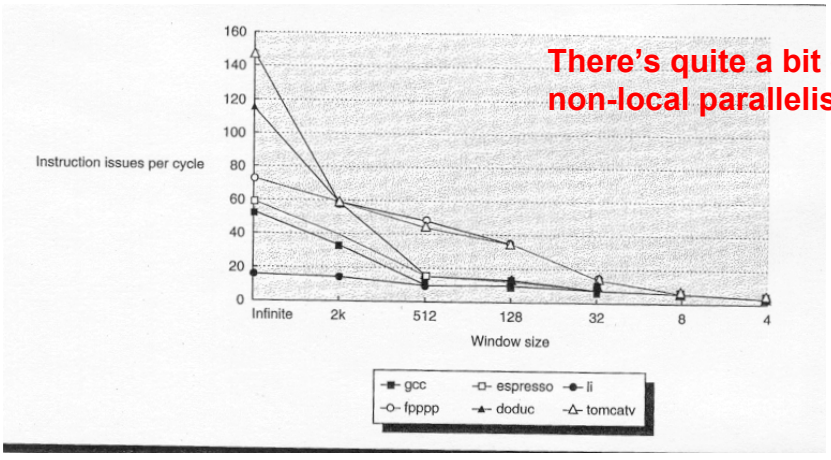
- Pentium … < 1 instruction/cycle retired
  - But low cycle time
  - Time= CPI × Instructions × CycleTime
- Not seen attempts to issue more than 4 instructions/cycle
  - Much less sustain retire or more than 4

Limit Effects

Superscalar

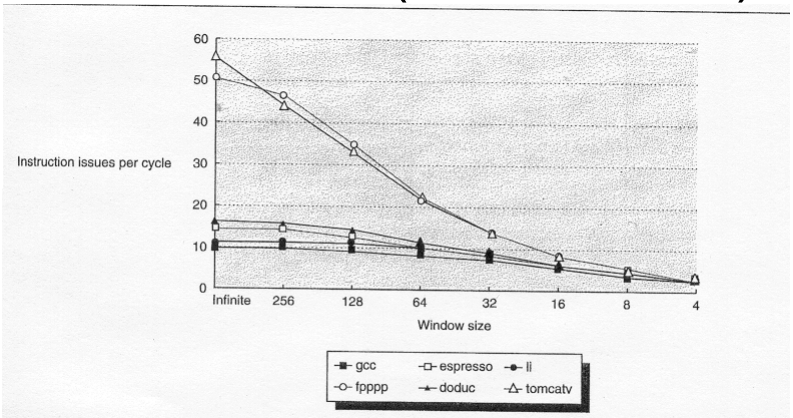


Window Size (unlimited issue)



[Hennessy and Patterson 4.39e2/3.36e3]

Window Size (Issue limited)



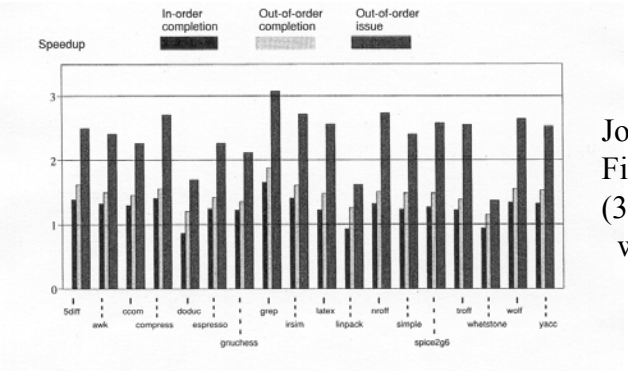
[64-issues Hennessy and Patterson 4.47e2/3.45e3]

Operation Organization

- Consider Tree-structured calculation
  - freedom in ordering
  - consider:
    - post-order traversal
    - by levels from leaves
  - where is parallelism?
  - Storage cost?

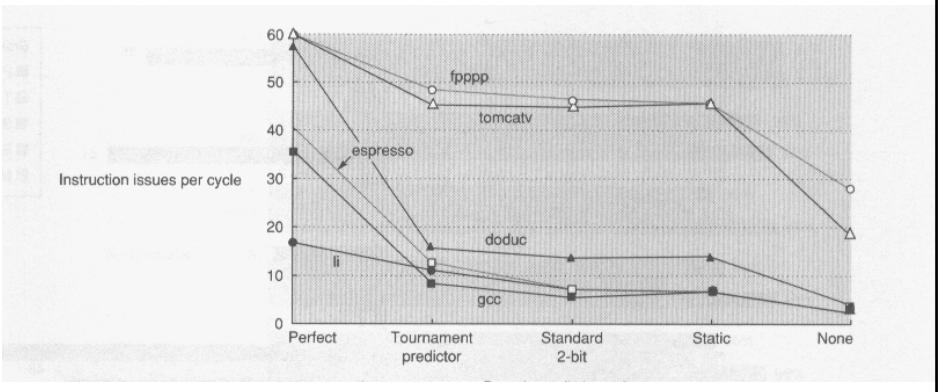
Window Size

- How many instructions forward do we look?
  - Only look at next = in-order issue



Johnson Fig. 3.9 (32 issue window?)

Branch Prediction



[Hennessey & Patterson Fig 3.38/e3]

Window Cost?

- No one before you in the window writes a value you need
- $Rsrc_i \neq Rdst_{i-1}$; $Rsrc_i \neq Rdst_{i-2}$;…
- $O(WS^2)$ comparisons

Cost?

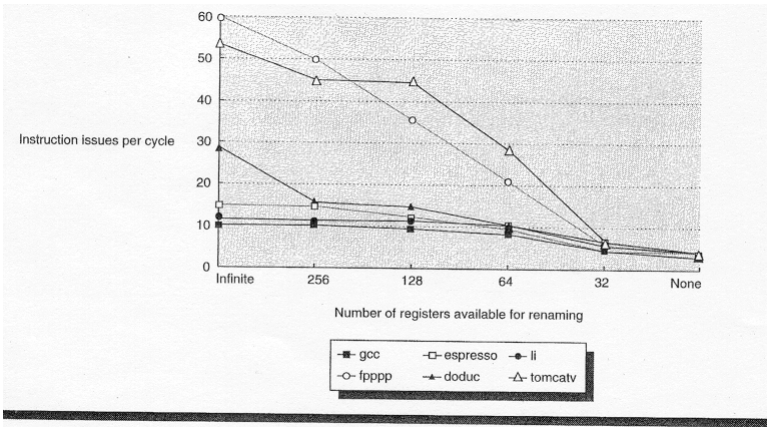
- Anecdotal [Farrell, Fischer JSSC v33n5]
  - DEC 20-instruction queue
  - 4 instruction issue
  - (80 physical registers)
  - $10mm^2$ in $0.35\mu m$ ($300M\lambda^2$+)
    - Compare:
      - 300 4-LUTs (w/ interconnect)
      - MIPS-X 32b CPU w/ 1KB memory = $68M\lambda^2$
  - 600 MHz = 1.6ns

Costs?

- Both DEC and "Quantifying" (also DEC)
  - appear to use a scoreboarded scheme to avoid
  - accept not issue until result computed?
- "Quantifyng" suggests:
  - wakeup time $\propto IW^2 \times WS^2$
    - but assuming quadratic wire delay in length
    - (never buffer wire)
  - but WS=F(IW)
  - certainly faster than linear time
  - $A \propto IW \times WS$

Registers

- How many virtual registers needed?



[Hennessy and Patterson 4.43e2/3.41e3]

Register Costs?

- First Order
  - area linear in number of registers
  - delay linear in number of registers
- Bank RF
  - maybe sublinear delay
  - at least square root number of registers
    - wire delay sqrt of area

RF and IW interaction

- Larger Issue (Decode)
  - want to read/retire more registers per cycle
  - RF ports = 3 IW [Op Rdst←Rsrc1,Rsrc2]
  - A $\propto$ ports $\times$ number
  - …and number of registers = F(IW)
  - A $\propto$ IW $\times$ F(IW)
- RF grows faster than linear

Bypass: Control

- Control comparison
  - every functional input (2 IW)
  - get input from
    - every pipestage (d) from issue produce to wb
    - for every result producer (>IW)
- Total comparisons: $d \times IW^2$

Bypass: Interconnect

- Linear layout
  - bypass span functional units and RF
  - physical RF grows with IW
    - read/write ports
    - more physical registers to support IW
  - FU bypass muxes grows with IW
- Consequently
  - width grows with IW
  - cycle grow with IW?

Bypass: Interconnect

- "Quantifying"
  - quadratic wire delay
  - (but asymptotically, we can buffer)
  - largest delay component calculated
    - (>1ns for IW=8) [180nm]
    - IW=8 about 5-6 times IW=4

Aliasing

- Do memory operations depend on one another?
- *E.g.*

  A[j+3]=x*x+y;

  Z=A[i-2]+A[i+2]
- Is A[i-2], A[i+2] another name for A[j+3]?
- *E.g.*

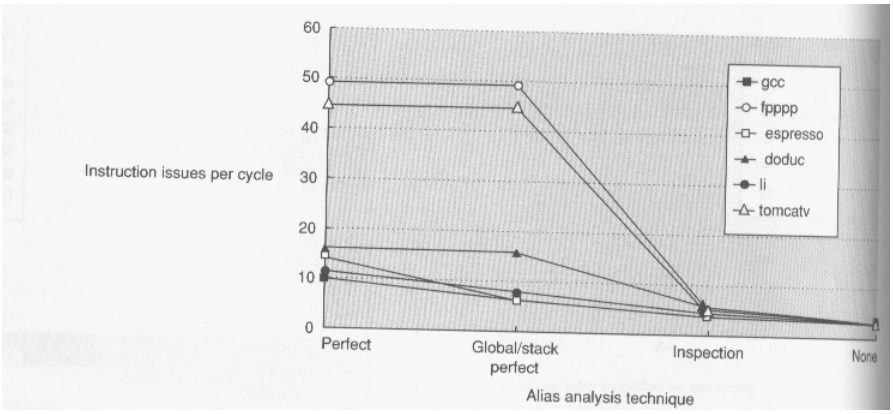
  *a++;

  *b+=3;

  *a++;

  d=*c+3;
- Are these operations all independent?
- Or do some name the same memory locaiton?

Aliasing



[Hennessey & Patterson Fig 3.43/e3]

…And now for something Completely Different

Different Solution

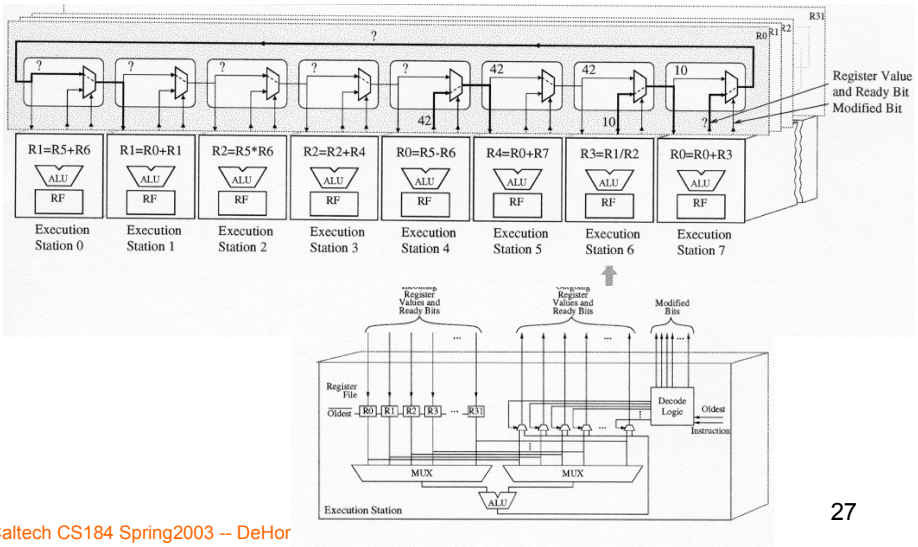
- These assume Number of Regs > IW
- If IW>R, different approach…

- From Henry, Kuszmaul, et. al.
  - ARVLSI'99
  - SPAA'99
  - ISCA'00

Consider Machine

- Each FU has a full RF
- Build network between FUs
  - use network to connect produce/consume
  - user register names to configure interconnect
- Signal data ready along network

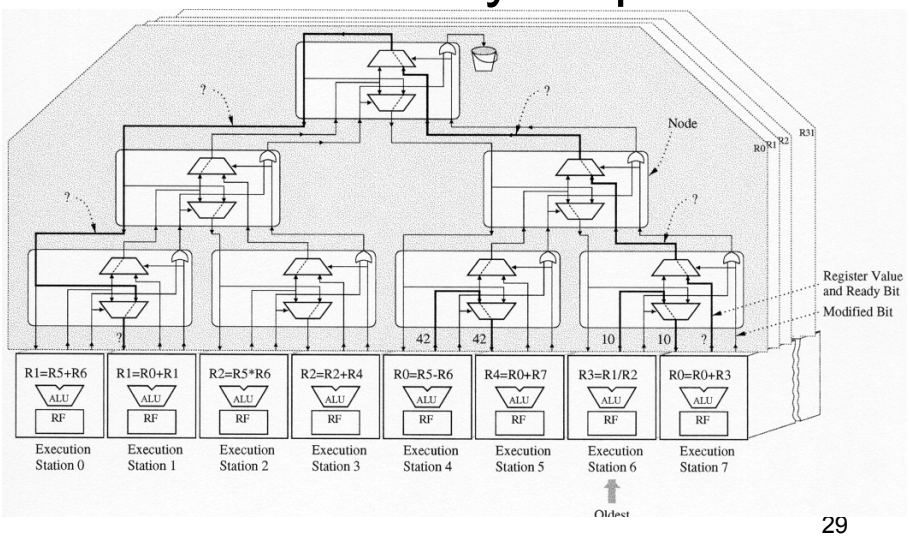
Ultrascalar: concept model



Ultrascalar concept

- Linear delay
- O(1) register cost / FU
- Complete renaming at each FU
  - different set of registers
  - so when say complete RF at each FU, that's only the logical registers

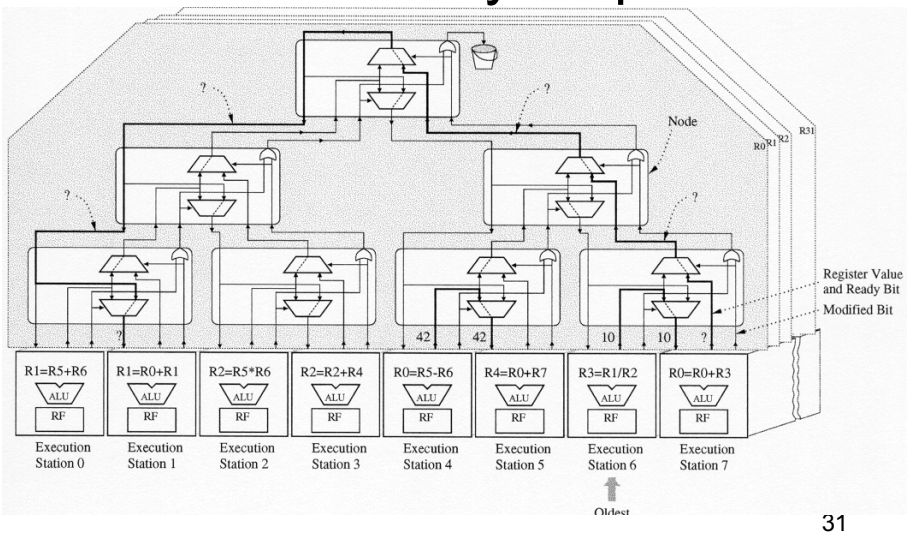
Ultrascalar: cyclic prefix



Parallel Prefix

- Basic idea is one we saw with adders
- An FU will either
  - produce a register (generate)
  - or transmit a register (propagate)
  - can do tree combining
    - pair of FUs will either both propagate or will generate
    - compute function by pair in one stage
    - recurse to next stage
    - get log-depth tree network connecting producer and consumer

Ultrascalar: cyclic prefix



Cyclic Prefix

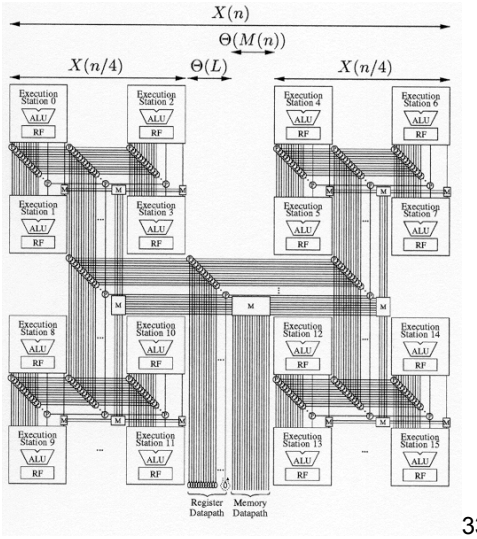
- Gets delay down to log(WS)
  - w/ linear layout, delay still linear
- Issue into, retire from Window in order
  - serves
    - rename
    - shared RF
    - issue
    - bypass
    - reorder

Ultrascalar: layout

Register paths not growing. (p=0 tree!)
Wide, but constant width

If Memory width <√n area goes as n

wire goes as √n



Ultrascalar: asymptotics

- Assume M(n)<O(√n)
  - Area ~ n×R²
  - Delay ~ (√n)×R
- Claim can do
  - Area ~ n×R
  - Delay ~ √(n×R)
- If memory grows faster, will dominate interconnect growth, hence area and delay
  - get extra term for memory growth (like Rent's Rule)

UltraScalar:

- 0.25 μm
- 128-window, 32 logical regs
- 64b ops ?
- 8 instruction fetch
- delays <2ns [0.25μm]
  - commit, wakeup, schedule
  - wire delay dominate logic
- area ~2Gλ^2 (not include datapath)

Solution for:

- Object/binary compatibility is paramount
- Performance is King
- Recompilation not an option
- Cost (area, energy) is no object

Friday

- …an alternative way to exploit ILP
- rely on compiler and feedback

- [**reminder:** no lecture Wednesday]

(Semi?) Big Ideas

- Good to look at
  - Extremes (what can this possibly do?)
  - Sensitivity (how important is this to…)
- Balance
- Size Matters
- Interconnect delay dominate
- As parameters grow
  - watch tradeoffs
  - widely different solutions prevail in different points in space (different asymptotes)