

CS184b: Computer Architecture [Single Threaded Architecture: abstractions, quantification, and optimizations]

Day10: February 6, 2000
VLIW

Today

- Trace Scheduling
- VLIW uArch
- Evidence for
- What it doesn't address

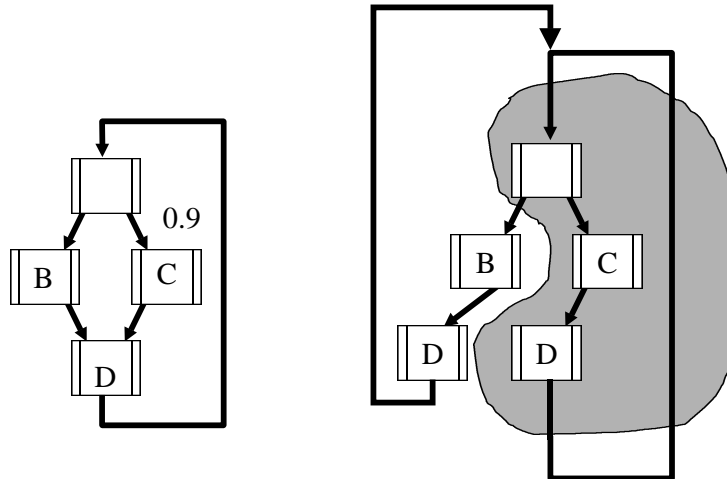
Problem

- Parallelism in Basic Block is limited
 - (recall average branch freq. Every 7-8 instrs)

Solution: Trace Scheduling

- Schedule likely sequences of code through branches
 - instrument code
 - capture execution frequency / branch probabilities
 - pick most common path through code
 - schedule as if that happens
 - add “patchup” code to handle *uncommon* case where exit trace
 - repeat for next most common case until done

Typical Example



Caltech CS184b Winter2001 -- DeHon

5

Solution Validity

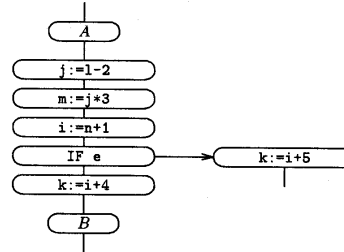
- Recall from Fisher/Predict paper
 - 50-150 instructions/mispredicted branch

Caltech CS184b Winter2001 -- DeHon

6

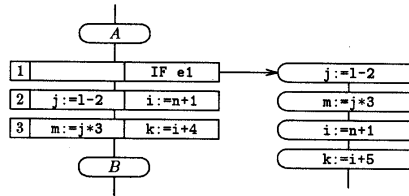
Trace Example

- Bulldog Fig 4.2



Bulldog: A Compiler for VLIW Architectures
 MIT Press 1986
 ACM Doctoral Dissertation Award 1985

1		IF e1
2	j:=1-2	i:=n+1
3	m:=j*3	k:=i+4

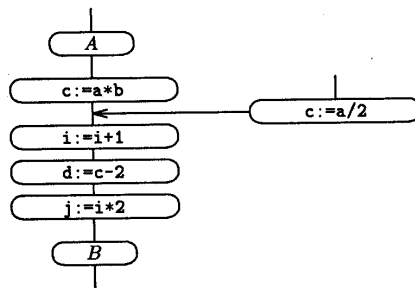


Caltech CS184b Winter2001 -- DeHon

7

Trace Join Example

Bulldog p61



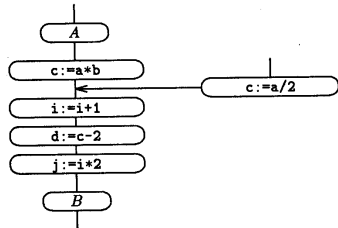
the code generator produced the following schedule:

1	c:=a*b	i:=i+1
2	d:=c-2	j:=i*2

Caltech CS184b Winter2001 -- DeHon

8

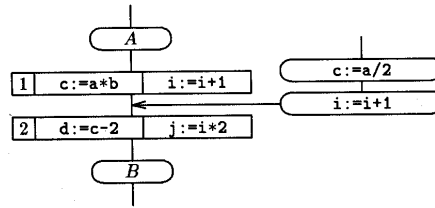
Trace Join Example



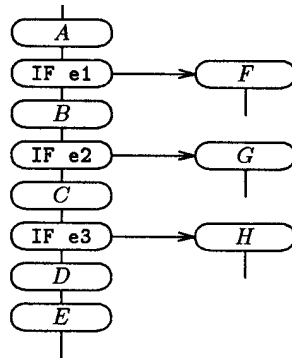
Bulldog p61-62

the code generator produced the

1	c:=a*b	i:=i+1
2	d:=c-2	j:=i*2



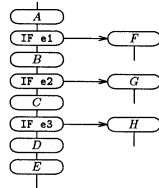
Trace Multi-Branch Example



Bulldog p69

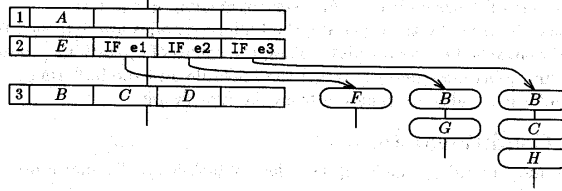
1	A			
2	E	IF e1	IF e2	IF e3
3	B	C	D	

Trace Multi-Branch Example



Bulldog p69-70

1	A			
2	E	IF e1	IF e2	IF e3
3	B	C	D	



Caltech CS184b Winter2001 -- DeHon

11

Trace Advantage

- Avoid fragmentation
 - can't fill issue slots because broken by branches
- Expose more parallelism
 - concurrent run things on different sides of branches
 - allow more global code motion (across branches)

Caltech CS184b Winter2001 -- DeHon

12

Machine

- Single PC/thread of control
- Wide instructions

- Branching
- Register File
- Memory Banking

Branching

- Allow multiple branches per “Instruction”
 - n-way branch
- N-tests + 1 fall-through
 - order in trace order
 - take first to succeed
- Encoding
 - single base address
 - branch to base+i
 - i is test which succeeded

Split Register File

- Each cluster has own RF
 - (register bank)
 - can have limited read/write bw
- Limited networking between clusters
 - explicit moves between clusters when results needed elsewhere

Memory Banks

- Separate Memory Banks
 - dispatch set of non-conflicting loads/stores, each to separate memory banks
 - trick is can compiler determine non-conflict
 - (do layout o avoid conflicts)
 - has to know won't conflict (for VLIW timing)

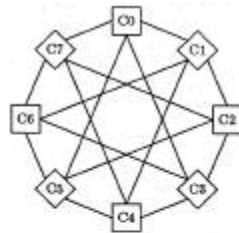
Memory Banks

- Avoid single memory bottleneck
- Avoid having to build n-ported memory
- Can make likelihood of conflict small
- Costs for crossbar between memory and consumers
- Arbitration required if can't statically schedule access pattern
- Hotspots/poor bank allocation can degrade performance

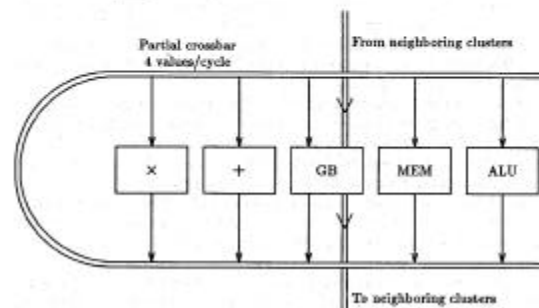
Caltech CS184b Winter2001 -- DeHon

17

ELI "Realistic"



Bulldog Fig 8.1



Caltech CS184b Win

18

Ellis Results

Program	Speed-up		Input size		Unrolling	
	Ideal	Real.	Ideal	Real.	Ideal	Real.
MATMUL	25.5	7.4	50 ²	50 ²	32	64
FFT	48.3	6.9	1024	1024	16	16
SOLVE	18.9	6.2	128 ²	128 ²	16	16/8
SVD	16.2	5.4	30 ²	128 ²	16/2	32/16
TRID1	2.7	.9	4096	4096	16	8
TRID2	3.8	1.2	4096	4096	16	8
TRID4	33.3	7.0	4096	4096	16/4	16/2
EOS	8.3	2.3	64 ²	64 ²	16	8
NEWZR	19.8	7.6	60 ²	49 ²	8	16
QK61	10.2	4.5	—	—	2	4
QUANC8	8.1	—	—	—	4	—
ZEROIN	3.5	—	—	—	4	—

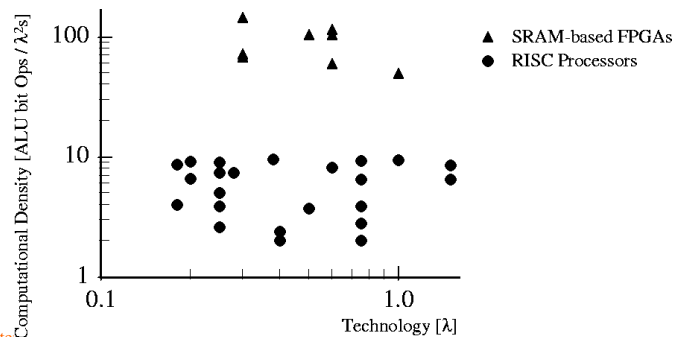
Caltech CS184b Winter2001 -- DeHon

Bulldog p242

19

Two CMOS VLIWs

- LIFE [ISSCC90] 23 ALU bops/ λ^2 s
- VIPER [JSSC93] 9.8



Caltech CS184b Winter2001 -- DeHon

20

What can/can't it do?

- Multiple Issue?
- Renaming?
- Branch prediction?
 - Static
 - dynamic
- Tolerate variable latency?
 - Memory
 - functional units

Scaling

- Issue
- Bypass
- Register File
- N-way branch
- Memory Banking
- RF-RF datapath

Scaling

- Linear Scaling
 - Issue
 - Bypass (only within cluster)
 - Register File (separate per cluster)
- Super linear
 - Memory Banking [(clusters)² ?]
 - RF-RF datapath ?
 - Unclear from small examples (and didn't study)

Scaling: N-way branch?

- Probably want to scale up branching with clusters (VLIW length)
- Use parallel prefix computation
 - depth goes as $\log(N)$
 - area can be linear

Scaling: Thoughts

- W/ on-chip memory
 - banks local to clusters (distributed memory)
 - can schedule operations on clusters close to memory?
 - Communicate data among clusters (like RF to RF transfers) if need non-local
 - How much interconnect needed?
 - What's the locality of data communication?
 - Recall interconnect richness study from last term

“Weaknesses”

- Binary Compatibility
 - lack thereof
- No “Architecture”
- Exceptions

Next Time

- EPIC
 - next generation VLIW evolution

Big Ideas

- Get better packing/performance scheduling large blocks
- Common case
- Feedback
 - (future like past)
 - discover common case
- Binding Time hoisting
 - Don't do at runtime what you can do at compile time
- Stable abstraction