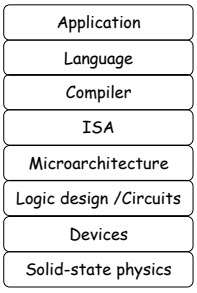
Lecture 4: Microarchitecture

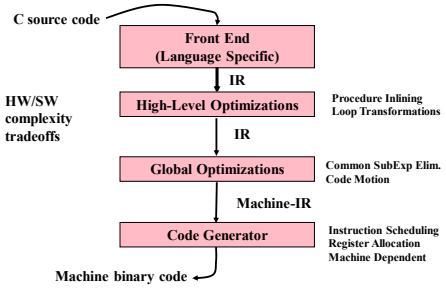
- Last Time
  - Overview of ISA components
  - MIPS ISA
  - ISA extensions, graphics processor ISAs and Philips trimedia ISA
- Today
  - Paper discussion
  - Microarchitecture
    - Components
    - Putting the components together

Slides courtesy of Stephen W. Keckler, UT-Austin

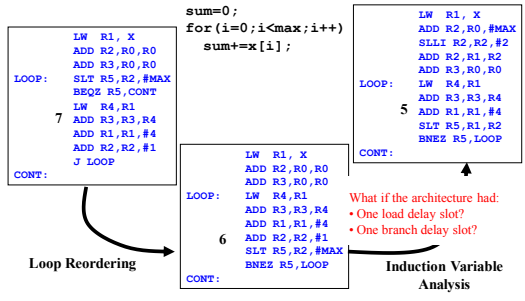
Layers of Abstraction



Role of the Optimizing Compiler



Example: Loop Optimization



Architect ⇒ Compiler Writer

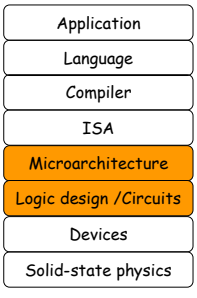
- Simplify, Simplify, Simplify
  - Feature difficult to use, it won't be used….Less is More!
- Regularity
  - Common set of formats, few special cases
- Primitive, not solutions
  - CALLS vs. Fast register moves
- Make performance tradeoffs simple
- Ultimately, the ISA will *not* be perfect

Compiler ⇒ Microarchitecture

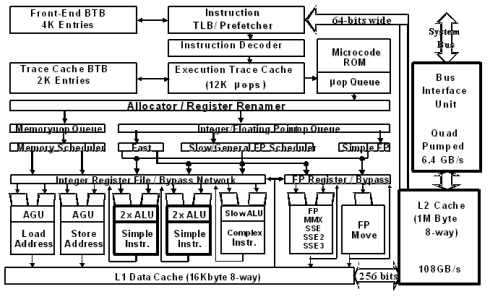
- Instruction Scheduling
  - Instruction Level Parallelism
- Resource Allocation
  - Registers (minimize spills/restores to and from memory)
- Memory optimizations
  - Cache conscious data organization
  - Code layout
- Etc……

Logic Design implements the Microarchitecture

- Digital behavior of chip
- Specifies:
  - Actual states (ISA visible and non-visible)
  - Transitions between states
- Consists of:
  - Combinational logic (ie. ALUs)
  - Sequential logic ("memory")
    - Registers
    - State machines



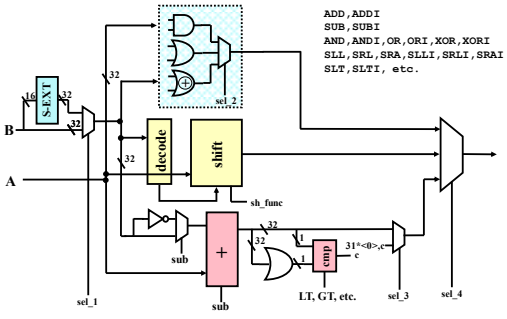
Pentium IV Microarchitecture Diagram



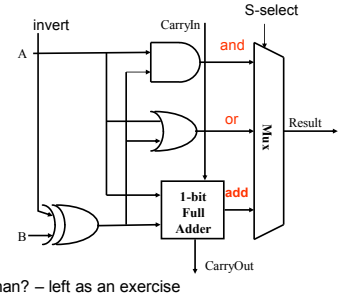
A simpler Microarchitecture Block Diagram

- Must implement ISA
  - What are basic components of ISA?
  - Opcode
    - Must provide functionality of each instruction
  - Storage
    - Register files
    - Memory
      - Caches, etc. are optimizations
  - Sequence the instructions
- Datapath vs. control path
  - ALU
  - Multiplier
  - Stall signals
  - Pipeline control

Combinational Logic: The ALU



Bit Slice Approach to ALU design

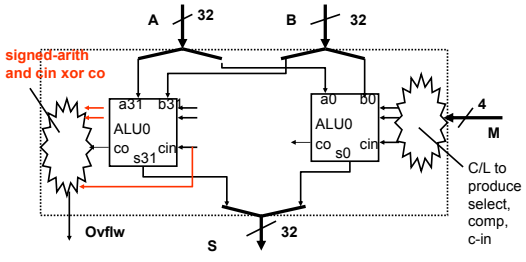


Set-less-than? – left as an exercise

Slide courtesy of D. Patterson

Bigger View

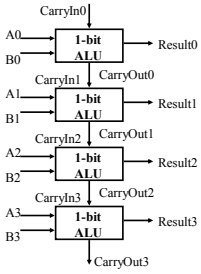
- LSB and MSB need to do a little extra



Slide courtesy of D. Patterson

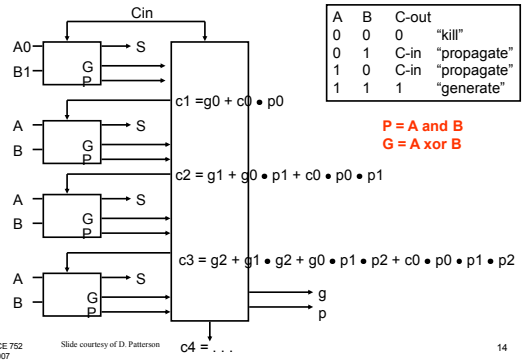
But What about Performance?

- Critical Path of n-bit Rippled-carry adder is n*CP



Design Trick: throw hardware at it

Carry Look Ahead (Design trick: peek)



A	B	C-out	
0	0	0	"kill"
0	1	C-in	"propagate"
1	0	C-in	"propagate"
1	1	1	"generate"

P = A and B
G = A xor B

Delay Analysis

- 16-bit ripple carry adder
  - $T = 16 * T_{FA}$
    $= 16 * 2$ gate delays $= 32$ gate delays
- 16-bit carry lookahead adder
  - $T = T_{C4}$
    $= 2 + \max(P_i, G_i) = 2 + T_{G0}$
    $= 2 + 2 + \max(p_i, g_i)$
    $= 5$ gate delays
  - Real designs use some nice transistor circuits for the carry lookahead chain
- In the limit – expand to full lookahead
  - T = 2 – but: a lot of logic, very wide fan-in gates
  - too costly

Wires

- Good
  - Short wires
  - Point-to-point
  - Unidirectional
- Bad
  - Long wires with many repeaters
  - Multidrop busses
  - Bidirectional

Review

- Circuit level
  - Transistors -> Logic gates
  - Storage element:
    - Flip-flop, dram cell, sram cell
- Logic design
  - Register files
    - Smaller capacity, more ports (5-6 ports not too painful)
  - SRAMs
    - Larger capacity, fewer ports
    - 2 ports ok for small arrays, 1 port for larger arrays
  - Content addressible memory (CAM)
    - Used for matching a bit-sequence to one stored in the CAM
  - Delay α capacity and ports
  - Area α capacity and ports^2
  - Power α capacity and ports

Review

- Logic design
  - ALU
  - Multipliers
  - more
- Tools of the microarchitect in place

Summary

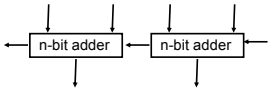
- Microarchitectural components
  - Control logic/datapath
  - Combinational circuits
  - Storage: flip-flops, registers, SRAMs, CAMs
- Next time
  - Pipelining/pipeline hazards
  - Pipeline discussion
- Homework 1 due
- Review:
  - Colwell, Instruction sets and beyond
  - Burger et al., Scaling to end of silicon…

Reference slides

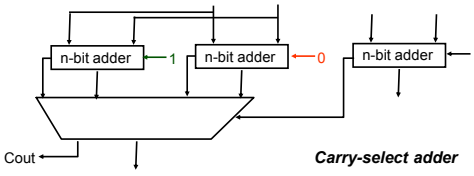
Other Techniques for Addition

Design Trick: Guess

T(2n) = 2*T(n)

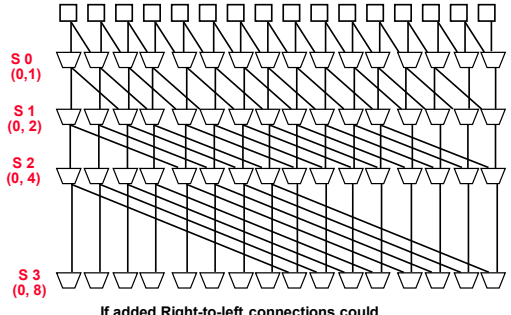


T(2n) = T(n) + T(mux)



Carry-select adder

General Shift Right Scheme using 16 bit example



If added Right-to-left connections could support Rotate (not in MIPS but found in ISAs)

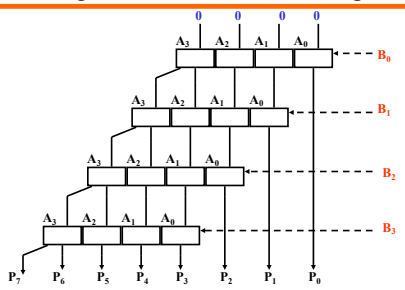
MULTIPLY (unsigned)

- Paper and pencil example (unsigned):

```
Multiplicand      1000
Multiplier        1001
                  1000
                 0000
                0000
               1000
Product        01001000
```

- m bits x n bits = m+n bit product
- Binary makes it easy:
  - 0 => place 0 (0 x multiplicand)
  - 1 => place a copy (1 x multiplicand)

Unsigned Combinational Multiplier



- Stage i accumulates $A * 2^i$ if $B_i == 1$
- Q: How much hardware for 32 bit multiplier? Critical path?

Observations on Multiplication

- Can speed up algorithm by doing multiple bits at a time, instead of just one (2, 4, etc)
  - See Booth encoding
- Multiplication algorithm is still slow
  - Each step is doing a full carry-propagate add
  - Can use carry save adders instead
  - Build a Wallace tree to combine the partial products
  - Single carry-propagate add instead
- What about division, square root, etc.