What was it like…. 

• 1983:
  – VLSI (maybe)
    No caches, persay if $ is no object
    Real compilers, with real compiler analysis
    not automatic thread extraction

• 1999:
  – Prior to intel buying Alpha technology
What is a VLIW machine?

- Single thread of control
  - As implemented, but...
- Some parallelism is exposed in instruction set encoding
Trace Scheduling

• Why?
  – Basic blocks are 6 or 7 instructions
  – 1.5 – 3 ILP within a basic block
• What is it?
  – Combine basic blocks into superbblocks
  – Schedule explicit ILP from superbblocks
  – Add fixups for when you're wrong
Trace Scheduling – How?

• What can you do wrong…
  – Register state
  – Memory state
  – Interrupts
    • Better NOT happen

• Better be able to undo
Branching

• ISA:
  – BEQ r0, r1, here; BLZ r4, r3, there; ADD ;
Pipelining?

- Delay slots
  - W/semantics
- Stall
- Predict
  - A little more challenging than non-VLIW
Memory bandwidth

- $\$
  - Banked
    - Multiple accesses to separate banks

- Cache
  - Larger instruction cache
  - Multi ported data cache
  - Feeding the beast
Memory Disambiguation – pointers?

- What makes it hard
  - Pointers
  - Control flow
  - Everything about OO programming
  - Dynamic scoping

- What makes it easy (Why did supercomputers focus on FORTRAN)
  - Global arrays
Procedures?

- Inline if source available
Vector vs. VLIW

• Fisher: Vector is crucifying difficult to program for
  – Data and computation has to be very regular
What information is only available dynamically?

- Cache misses
- Branches are a little easier
Software pipelining

- For(x=0; x<j; x++) {
    r[x] = a[x] + b[x];
    if(r[x] > 255)
        r[x] = 255;
}