What are the benefits of VLIW?

- Compiler can describe parallelism to the HW
- Philosophy that encourages complexity in the SW instead of the HW

What are the downsides to VLIW?

- Compilers must use profile-driven optimizations
  - Bad-profiles can mess-up performance
- Sometimes there isn't a good common-case trace
  - Not a good fit for small functions
- Not all ILP is statically identifiable

Branches

- BEQ here; BLT overthere; JUMP somewhere; fall-through

Memory

- STORE reg1, @(reg2); ... ; LOAD reg3, @(reg4)

Forwarding/Bypassing

- ADD r1, r2 -> r3 ; XOR x1, x2 -> r4
- SUB r3, r4 -> r5 ; NAND r3, r1 -> r6

Why did I ask you to read the GPA paper?

- Both about letting the compiler find parallelism
- Static scheduling:
  - GPA:Static placement / Dynamic scheduling
- Point-to-point communication between FU's addresses bypassing problem

How are branches handled?

What about memory accesses?

Where might you want to use VLIW?