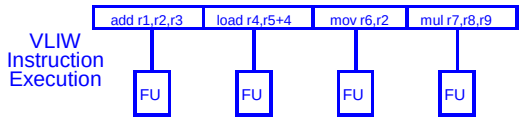
Very Long Instruction Word (VLIW) Architectures

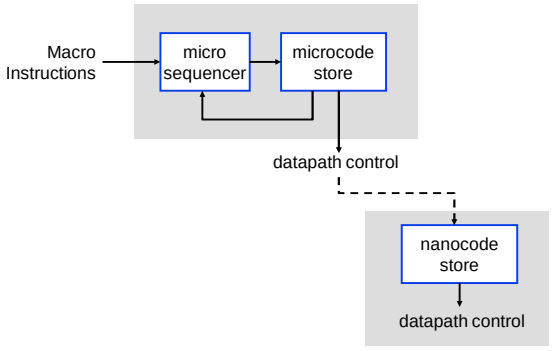
55:132/22C:160
High Performance Computer Architecture

What Is VLIW?

- VLIW hardware is simple and straightforward,
- VLIW separately directs each functional unit

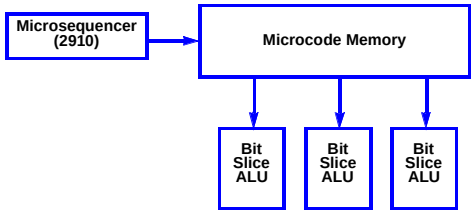


Historical Perspective: Microcoding, nanocoding (and RISC)



Horizontal Microcode and VLIW

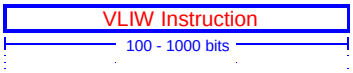
- A generation of high-performance, application-specific computers relied on *horizontally* microprogrammed computing engines.



- Aggressive (but tedious) hand programming at the microcode level provided performance well above sequential processors.

Principles of VLIW Operation

- Statically scheduled ILP architecture.
- Wide instructions specify several independent simple operations.



- Multiple functional units execute all of the operations in an instruction concurrently, providing fine-grain parallelism within each instruction
- Instructions directly control the hardware with no interpretation and minimal decoding.
- A powerful optimizing compiler is responsible for locating and extracting ILP from the program and for scheduling operations to exploit the available parallel resources

The processor does not make any run-time control decisions below the program level

Formal VLIW Models

- Josh Fisher proposed the first VLIW machine at Yale (1983)
- Fisher's *Trace Scheduling* algorithm for microcode compaction could exploit more ILP than any existing processor could provide.
- The ELI-512 was to provide massive resources to a single instruction stream
  - 16 processing clusters- multiple functional units/cluster.
  - partial crossbar interconnect.
  - multiple memory banks.
  - attached processor – no I/O, no operating system.
- Later VLIW models became increasingly more regular
  - Compiler complexity was a greater issue than originally envisioned

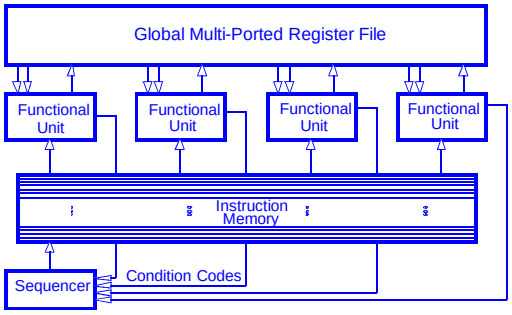
Ideal Models for VLIW Machines

- Almost all VLIW research has been based upon an ideal processor model.
- This is primarily motivated by compiler algorithm developers to simplify scheduling algorithms and compiler data structures.
  - This model includes:
    - Multiple universal functional units
    - Single-cycle global register file

    and often:
    - Single-cycle execution
    - Unrestricted, Multi-ported memory
    - Multi-way branching

    and sometimes:
    - Unlimited resources (Functional units, registers, etc.)

VLIW Execution Characteristics



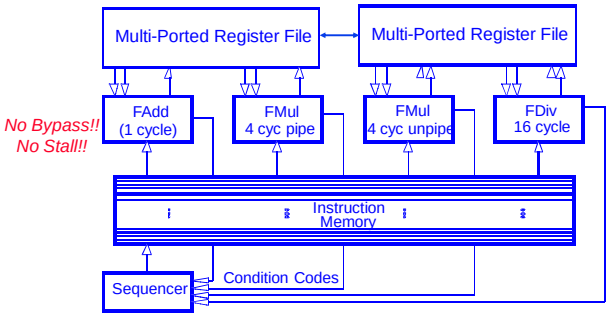
Basic VLIW architectures are a generalized form of horizontally microprogrammed machines

VLIW Design Issues

- Unresolved design issues
  - The best functional unit mix
  - Register file and interconnect topology
  - Memory system design
  - Best instruction format
- Many questions could be answered through experimental research
  - Difficult - needs effective retargetable compilers
- Compatibility issues still limit interest in general-purpose VLIW technology

However, VLIW may be the only way to build 8-16 operation/cycle machines.

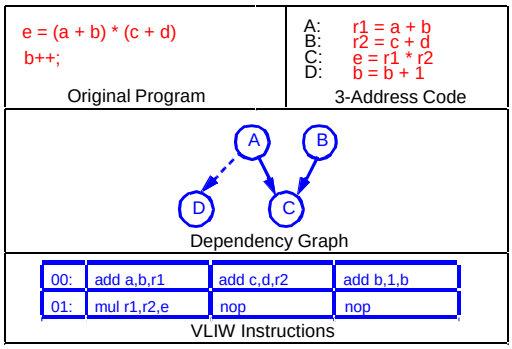
Realistic VLIW Datapath



Scheduling for Fine-Grain Parallelism

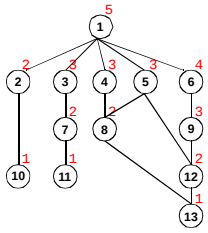
- The program is translated into primitive RISC-style (three address) operations
- Dataflow analysis is used to derive an operation precedence graph from a portion of the original program
- Operations which are independent can be scheduled to execute concurrently contingent upon the availability of resources
- The compiler manipulates the precedence graph through a variety of semantic-preserving transformations to expose additional parallelism

Example



VLIW List Scheduling

- Assign Priorities
- Compute Data Ready List - all operations whose predecessors have been scheduled.
- Select from DRL in priority order while checking resource constraints
- Add newly ready operations to DRL and repeat for next instruction



4-wide VLIW				Data Ready List
1				{1}
6	3	4	5	{2,3,4,5,6}
9	2	7	8	{2,7,8,9}
12	10	11		{10,11,12}
13				{13}

Enabling Technologies for VLIW

- VLIW Architectures attempt to achieve high performance through the combination of a number of key enabling *hardware* and *software* technologies.
  - Optimizing Schedulers (compilers)
  - Static Branch Prediction
  - Symbolic Memory Disambiguation
  - Predicated Execution
  - (Software) Speculative Execution
  - Program Compression

Strengths of VLIW Technology

- Parallelism can be exploited at the instruction level
  - Available in both vectorizable and sequential programs.
- Hardware is regular and straightforward
  - Most hardware is in the datapath performing useful computations.
  - Instruction issue costs scale approximately linearly

  Potentially very high clock rate
- Architecture is "*Compiler Friendly*"
  - Implementation is completely exposed - 0 layer of interpretation
  - Compile time information is easily propagated to run time.
- Exceptions and interrupts are easily managed
- Run-time behavior is highly predictable
  - Allows real-time applications.
  - Greater potential for code optimization.

Weaknesses of VLIW Technology

- No object code compatibility between generations
- Program size is large (explicit NOPs)

  Multiflow machines predated "dynamic memory compression" by encoding NOPs in the instruction memory
- Compilers are extremely complex
  - Assembly code is almost impossible
- Difficulties with variable memory latencies (caching)
- VLIW memory systems can be very complex
  - Simple memory systems may provide very low performance
  - Program controlled multi-layer, multi-banked memory
- Parallelism is underutilized for some algorithms.

VLIW vs. Superscalar

Attributes	Superscalar	VLIW
Multiple instructions/cycle	yes	yes
Multiple operations/instruction	no	yes
Instruction stream parsing	yes	no
Run-time analysis of register dependencies	yes	no
Run-time analysis of memory dependencies	maybe	occasionally
Runtime instruction reordering	yes (Resv. Stations)	no
Runtime register allocation	yes (renaming)	maybe (iteration frames)

Real VLIW Machines

- VLIW Minisupercomputers/Superminicomputers:
  - Multiflow TRACE 7/300, 14/300, 28/300 *[Josh Fisher]*
  - Multiflow TRACE /500 *[Bob Colwell]*
  - Cydrome Cydra 5 *[Bob Rau]*
  - IBM Yorktown VLIW Computer (research machine)
- Single-Chip VLIW Processors:
  - Intel iWarp, Philip's LIFE Chips (research)
- Single-Chip VLIW Media (through-put) Processors:
  - Trimedia, Chromatic, Micro-Unity
- DSP Processors (TI TMS320C6x)
- **Intel/HP EPIC IA-64 (Explicitly Parallel Instruction Comp.)**
- Transmeta Crusoe (x86 on VLIW??)
- Sun MAJC (Microarchitecture for Java Computing)