VLIW PROCESSORS

Department of E &TC, MITCOE, Pune

Introduction

- **Very long instruction word** or **VLIW** refers to a processor architecture designed to take advantage of instruction level parallelism
  - Instruction of a VLIW processor consists of multiple independent operations grouped together.
  - There are Multiple Independent Functional Units in VLIW processor architecture.
  - Each operation in the instruction is aligned to a functional unit.
  - All functional units share the use of a common large register file.
- This type of processor architecture is intended to allow higher performance without the inherent complexity of some other approaches.

Different Approaches

Other approaches to improving performance in processor architectures :

- **Pipelining**

  Breaking up instructions into sub-steps so that instructions can be executed partially at the same time

- **Superscalar architectures**

  Dispatching individual instructions to be executed completely independently in different parts of the processor

- **Out-of-order execution**

  Executing instructions in an order different from the program

Instruction Level Parallelism (ILP)

- **Instruction-level parallelism** (ILP) is a measure of how many of the operations in a computer program can be performed simultaneously.
- The overlap among instructions is called instruction level parallelism.
- Ordinary programs are typically written under a sequential execution model where instructions execute one after the other and in the order specified by the programmer.
- Goal of compiler and processor designers implementing ILP is to identify and take advantage of as much ILP as possible.

What is ILP? (Example)

Consider the following program:

```
op 1   e = a + b
op2   f = c + d
op3   m = e * f
```

- Operation 3 depends on the results of operations 1 and 2, so it cannot be calculated until both of them are completed
- However, operations 1 and 2 do not depend on any other operation, so they can be calculated simultaneously
- If we assume that each operation can be completed in one unit of time then these three instructions can be completed in a total of two units of time
- giving an ILP of 3/2.

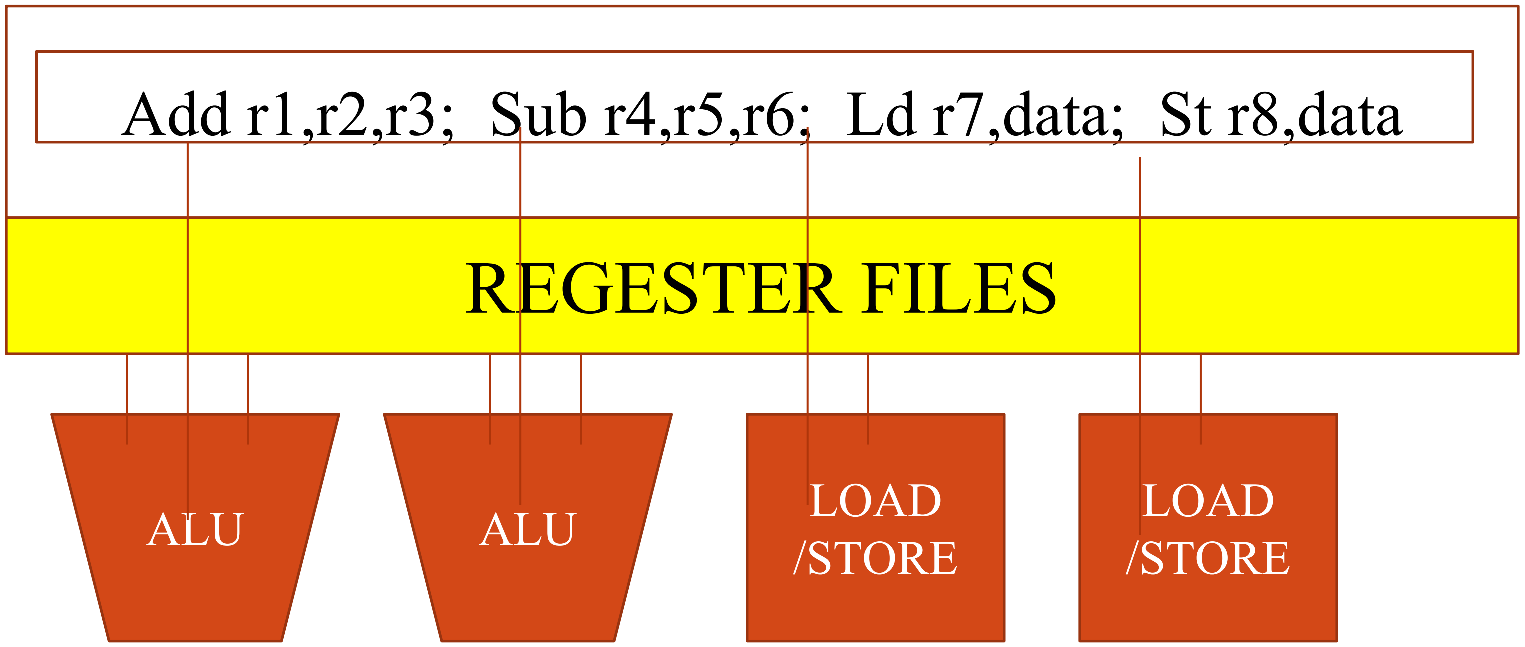
VLIW Compiler

- Compiler is responsible for static scheduling of instructions in VLIW processor.
- Compiler finds out which operations can be executed in parallel in the program.
- It groups together these operations in single instruction which is the very large instruction word.
- Compiler ensures that an operation is not issued before its operands are ready.

VLIW Instruction

- One VLIW instruction word encodes multiple operations which allows them to be initiated in a single clock cycle.
- The operands and the operation to be performed by the various functional units are specified in the instruction itself.
- One instruction encodes at least one operation for each execution unit of the device.
- So length of the instruction increases with the number of execution units
- To accommodate these operation fields, VLIW instructions are usually at least 64 bits wide, and on some architectures are much wider up to 1024 bits.

VLIW Instruction



ILP in VLIW

- Consider the computation of $y = a1x1 + a2x2 + a3x3$

On a sequential processor	**On the VLIW processor with 2 load/store units, 1 multiply unit and 1 add unit**
cycle 1: load a1 *cycle 2: load x1* *cycle 3: load a2* *cycle 4: load x2* *cycle 5: multiply z1 a1 x1* *cycle 6: multiply z2 a2 x2* *cycle 7: add y z1 z2* *cycle 8: load a3* *cycle 9: load x3* *cycle 10: multiply z1 a3 x3* *cycle 11: add y y z2*	*cycle 1: load a1* *load x1* *cycle 2: load a2* *load x2* *Multiply z1 a1 x1* *cycle 3: load a3* *load x3* *Multiply z2 a2 x2* *cycle 4: multiply z3 a3 x3* *add y z1 z2* *cycle 5: add y y z3*
requires 11 cycles.	requires 5 cycles.

Block Diagram

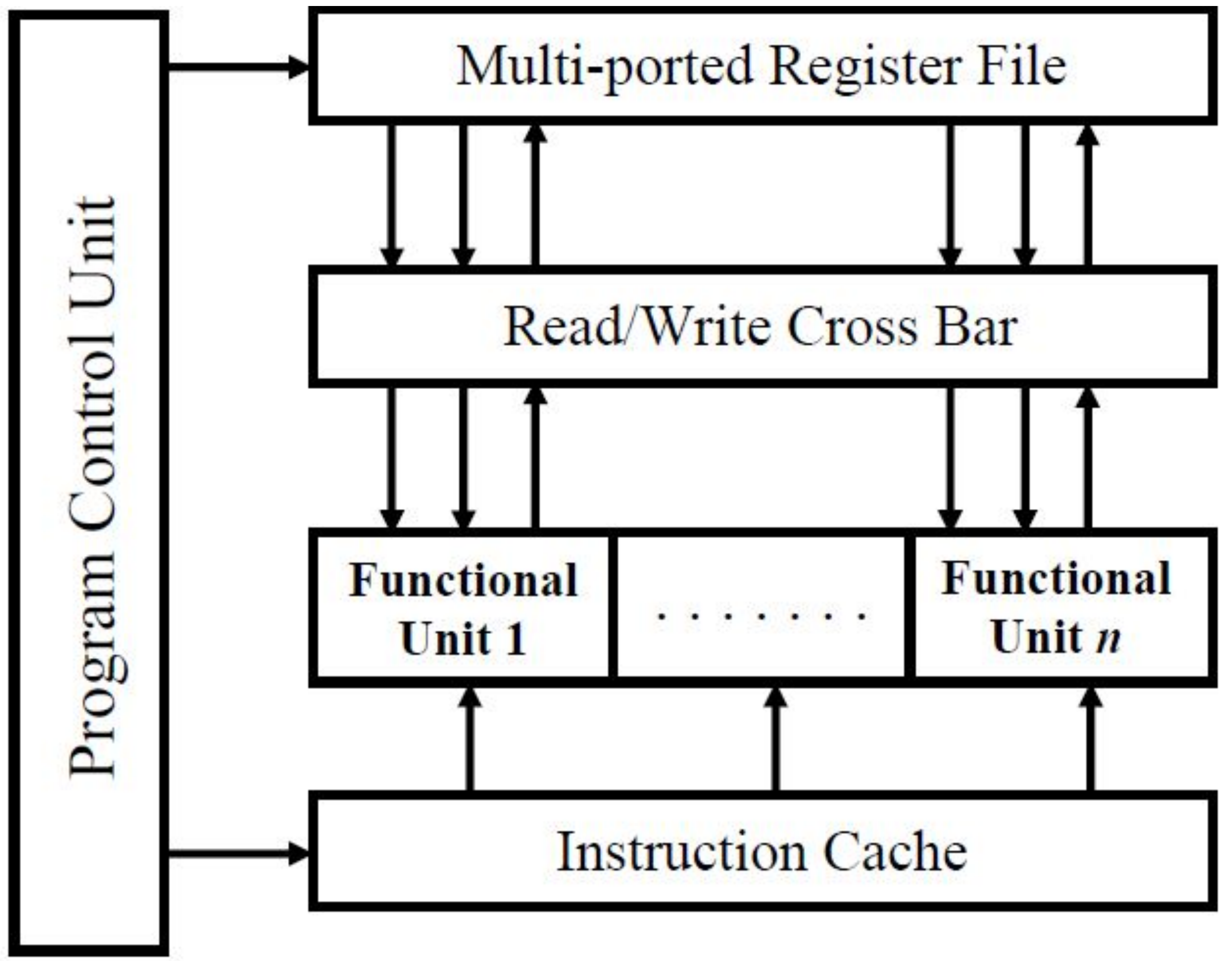
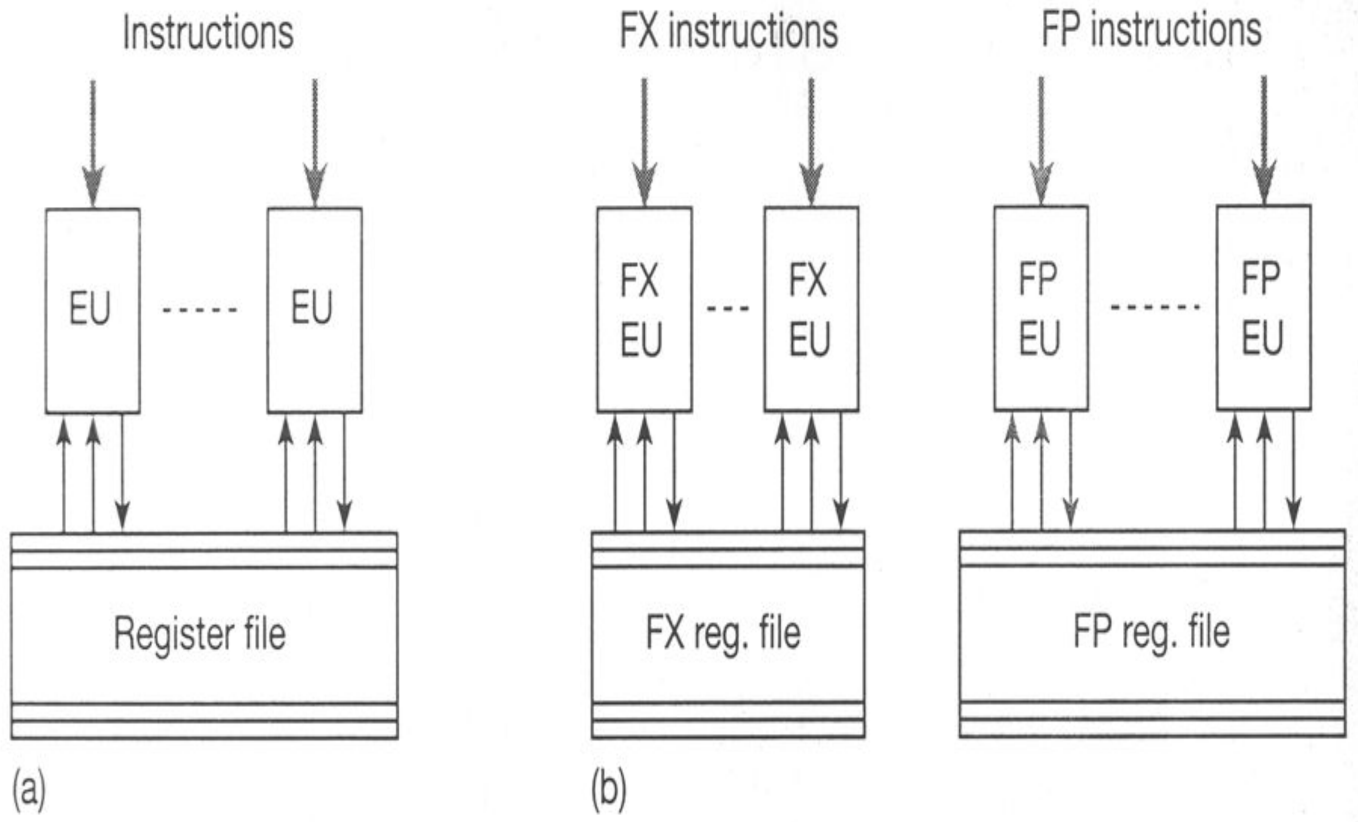


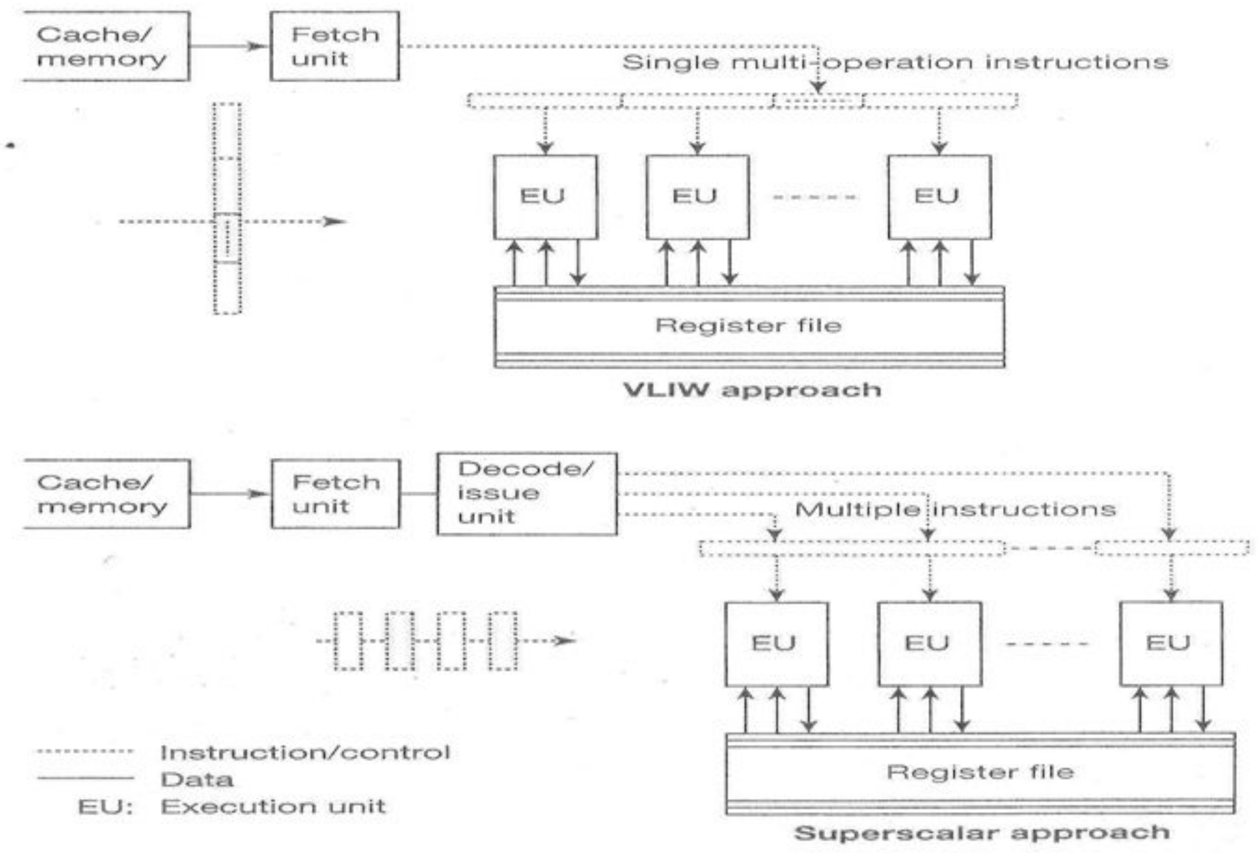
Diagram (Conceptual Instruction Execution)



Working

- Long instruction words are fetched from the memory
- A common multi-ported register file for fetching the operands and storing the results.
- Parallel random access to the register file is possible through the read/write cross bar.
- Execution in the functional units is carried out concurrently with the load/store operation of data between RAM and the register file.
- One or multiple register files for FX and FP data.
- Rely on compiler to find parallelism and schedule dependency free program code.

Difference Between VLIW & Superscalar Architecture



VLIW vs. Superscalar Architecture

- **Instruction formulation**
  - *Superscalar:*
    - Receive conventional instructions conceived for sequential processors.
  - *VLIW:*
    - Receive long instruction words, each comprising a field (or opcode) for each execution unit.
    - Instruction word length depends number of execution units and code length to control each unit (such as opcode length, registers).
    - Typical word length is 64 – 1024 bits, much longer than conventional machine word length.

VLIW vs. Superscalar Architecture

- **Instruction scheduling**
  - *Superscalar:*
    - Done dynamically at run-time by the hardware.
    - Data dependency is checked and resolved in hardware.
    - Need a look ahead hardware window for instruction fetch.
  - *VLIW:*
    - Done statically at compile time by compiler.
    - Data dependency is checked by compiler.
    - In case of un-filled opcodes in a VLIW, memory space and instruction bandwidth are wasted.

Comparison: CISC, RISC, VLIW

ARCHITECTURE CHARACTERISTIC	CISC	RISC	VLIW
Instruction Size	Varies	One size, usually 32 bits	One size
Instruction Semantics	Varies from simple to complex; possibly many dependent operations per instruction	Almost always one simple operation	Many simple, independent operations
Registers	Few, sometimes special	Many, general-purpose	Many, general-purpose
Hardware Design	Exploit microcode implementations	Exploit implementations with one pipeline and & no microcode	Exploit implementations with multiple pipelines, no microcode & no complex dispatch logic

Advantages of VLIW

- Dependencies are determined by compiler and used to schedule according to function unit latencies .
- Function units are assigned by compiler and correspond to the position within the instruction packet.
- Reduces hardware complexity.
  - Tasks such as decoding, data dependency detection, instruction issues etc. becoming simple.
  - Ensures potentially higher Clock Rate.
  - Ensures Low power consumption

Disadvantages of VLIW

- Higher complexity of the compiler
- Compatibility across implementations : Compiler optimization needs to consider technology dependent parameters such as latencies and load-use time of cache.
- Unscheduled events (e.g. cache miss) stall entire processor .
- Code density: In case of un-filled opcodes in a VLIW, memory space and instruction bandwidth are wasted i.e. low slot utilization.
- Code expansion: Causes high power consumption

Applications

- VLIW architecture is suitable for Digital Signal Processing applications.
- Processing of media data like compression/decompression of Image and speech data.

Examples of VLIW processor

- VLIW Mini supercomputers:

  Multiflow TRACE 7/300, 14/300, 28/300

  Multiflow TRACE /500

  Cydrome Cydra 5

  IBM Yorktown VLIW Computer
- Single-Chip VLIW Processors:

  Intel iWarp, Philip's LIFE Chips
- Single-Chip VLIW Media (through-put) Processors:

  Trimedia, Chromatic, Micro-Unity
- DSP Processors (TI TMS320C6x)