

# VLIW Processors

P. K. Roy

Asst. Professor

Siliguri Institute of Technology

# Basic Working Principles of VLIW

- **V**ery **L**ong **I**nstruction **W**ord (typical word length from 64 bits to 1 Kbits)
- Multiple independent Functional Units
- Instruction consists of multiple independent instructions (provides multiple operations in a single instruction).
- Each of them is aligned to a functional unit
- Latencies are fixed
  - Architecturally visible
- Compiler packs instructions into a VLIW also schedules all hardware resources
- Entire VLIW issues as a single unit

# Basic Working Principles of VLIW

- Aim at speeding up computation by exploiting instruction-level parallelism.
- Same hardware core as superscalar processors, having multiple execution units (EUs) working in parallel.
- All operations in an instruction are executed in a lock-step mode.
- Hardware does not check anything
- Software has to schedule so that all works
- Result: ILP with simple hardware
  - compact, fast hardware control
  - fast clock

# Basic Working Principles of VLIW

Instruction format

ALU1	ALU2	MEM1	control
------	------	------	---------

Program order and execution order

ALU1	ALU2	MEM1	control
ALU1	ALU2	MEM1	control
ALU1	ALU2	MEM1	control

Multiple functional units execute all operations in an instruction concurrently, providing fine-grain parallelism within instruction.

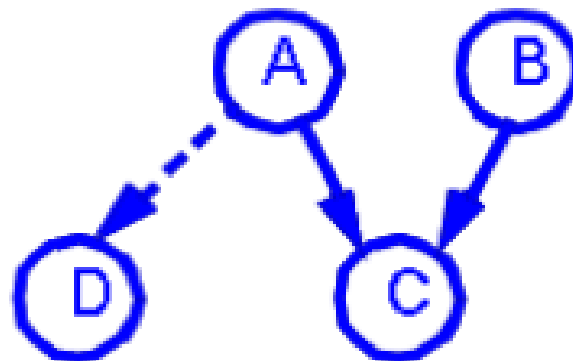
# Example

$e = (a + b) * (c + d)$   
 $b++;$

Original Program

A:  $r1 = a + b$   
B:  $r2 = c + d$   
C:  $e = r1 * r2$   
D:  $b = b + 1$


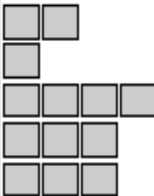
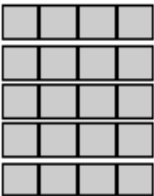
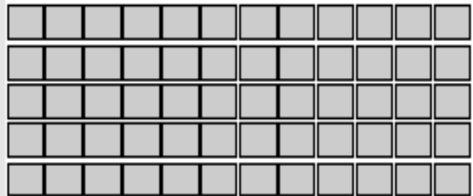
3-Address Code



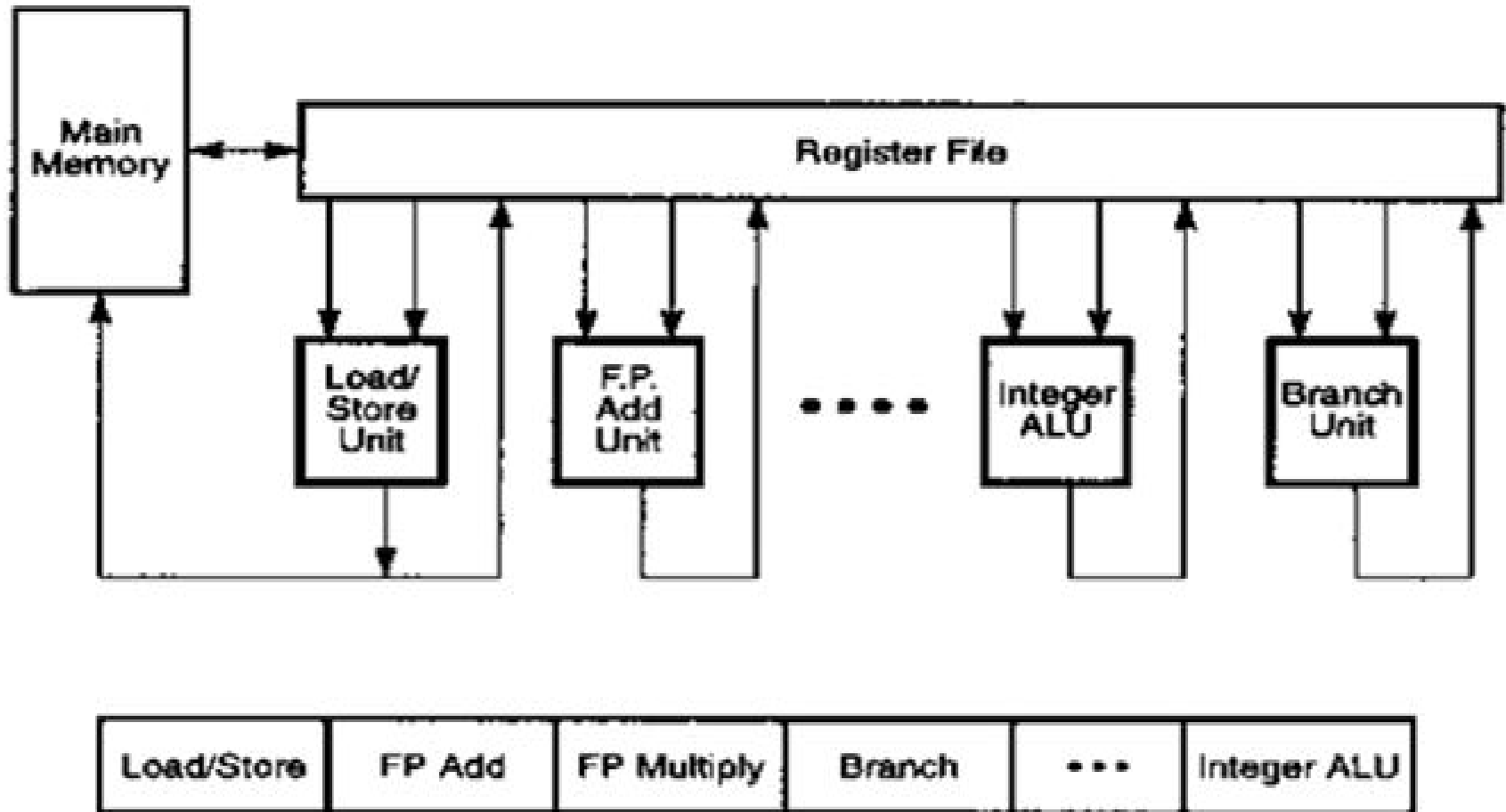
Dependency Graph

00:	add a,b,r1	add c,d,r2	add b,1,b
01:	mul r1,r2,e	nop	nop

VLIW Instructions

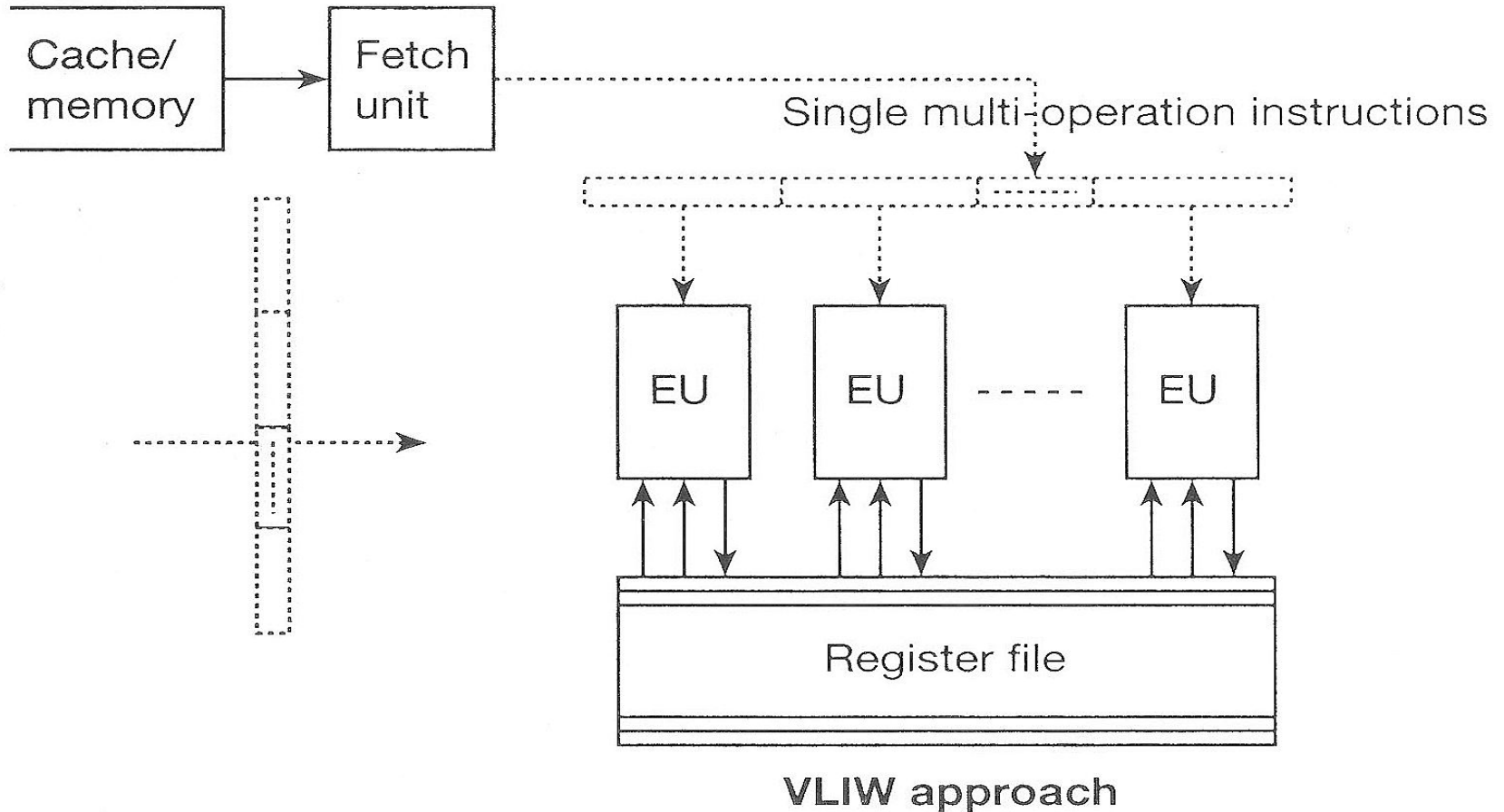
ARCHITECTURE CHARACTERISTIC	CISC	RISC	VLIW
INSTRUCTION SIZE	Varies	One size, usually 32 bits	One size
INSTRUCTION FORMAT	Field placement varies	Regular, consistent placement of fields	Regular, consistent placement of fields
INSTRUCTION SEMANTICS	Varies from simple to complex; possibly many dependent operations per instruction	Almost always one simple operation	Many simple, independent operations
REGISTERS	Few, sometimes special	Many, general-purpose	Many, general-purpose
MEMORY REFERENCES	Bundled with operations in many different types of instructions	Not bundled with operations, i.e., load/store architecture	Not bundled with operations, i.e., load/store architecture
HARDWARE DESIGN FOCUS	Exploit microcoded implementations	Exploit implementations with one pipeline and no microcode	Exploit implementations with multiple pipelines, no microcode & no complex dispatch logic
PICTURE OF FIVE TYPICAL INSTRUCTIONS  = 1 BYTE			

# VLIW Processor (1)



A typical VLIW processor and instruction format

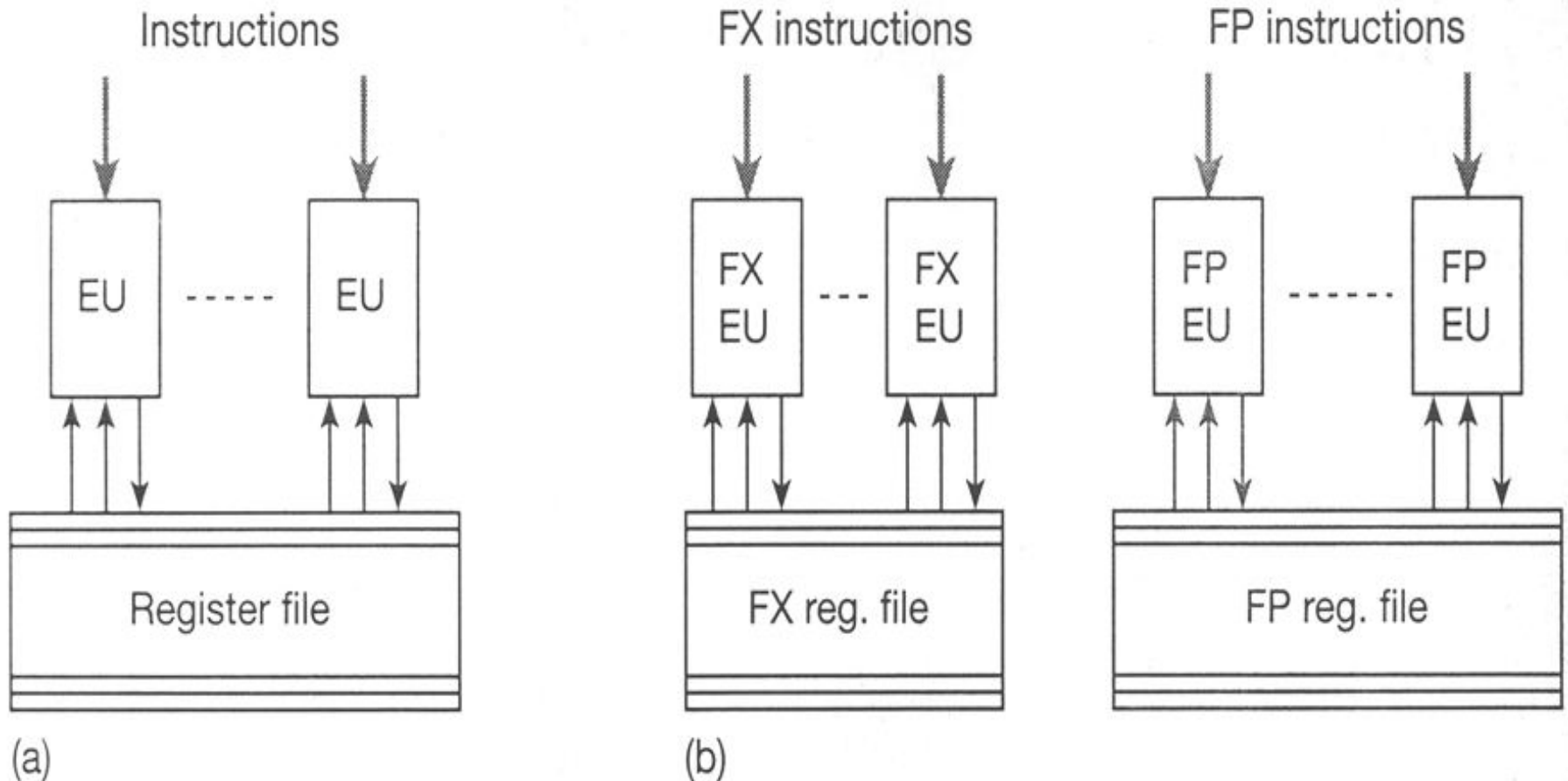
# VLIW Processor (2)



*Self Review: The Trace 200 Family (Karsuk)*

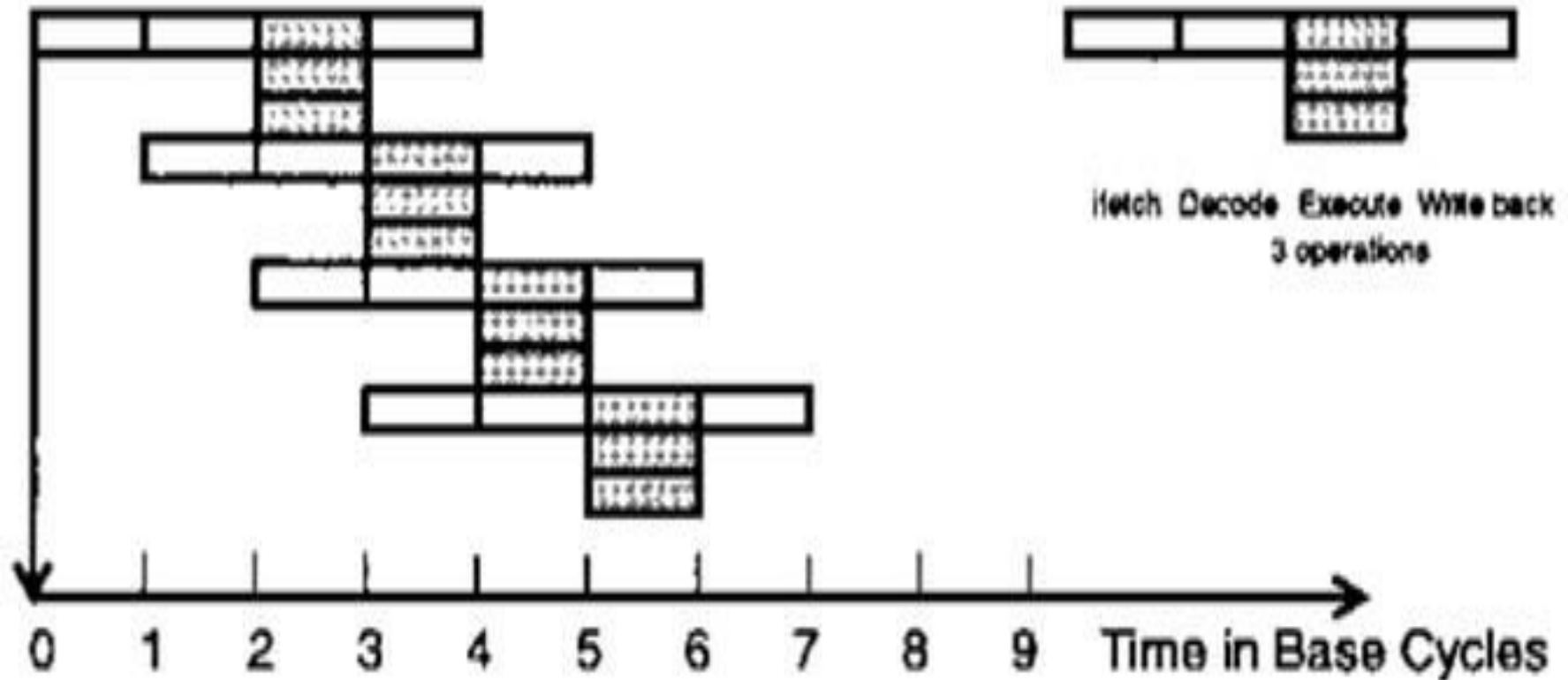


# Register File Structure for VLIW



Common basic structure of superscalar and VLIW architectures (a) Unified register file; (b) split register file.

# VLIW Execution

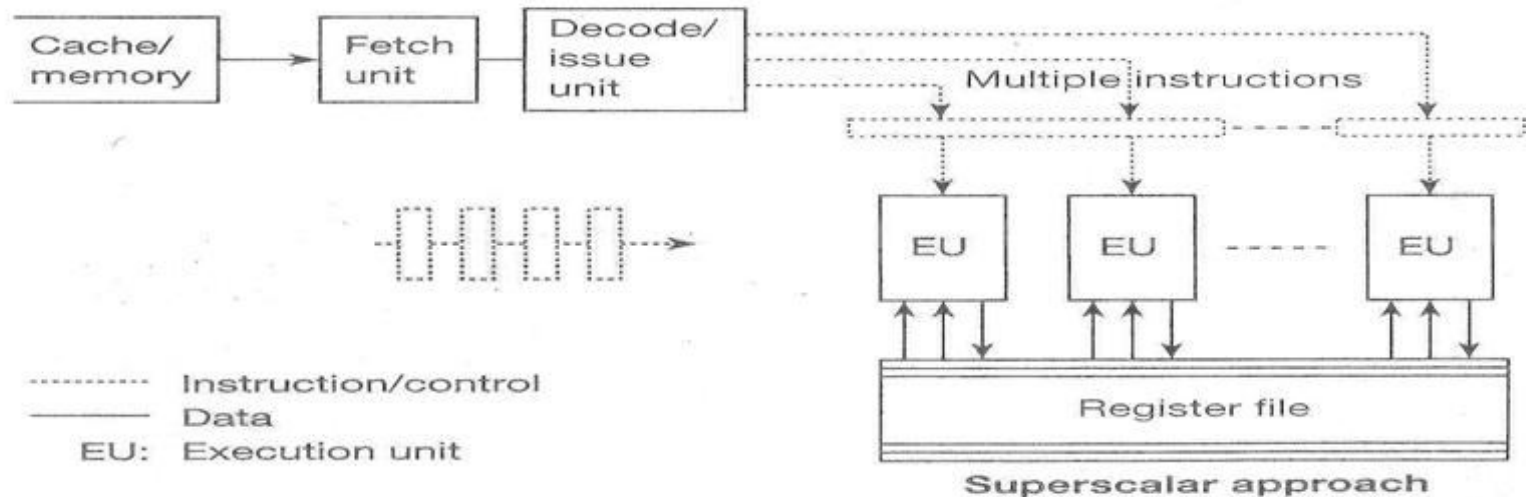
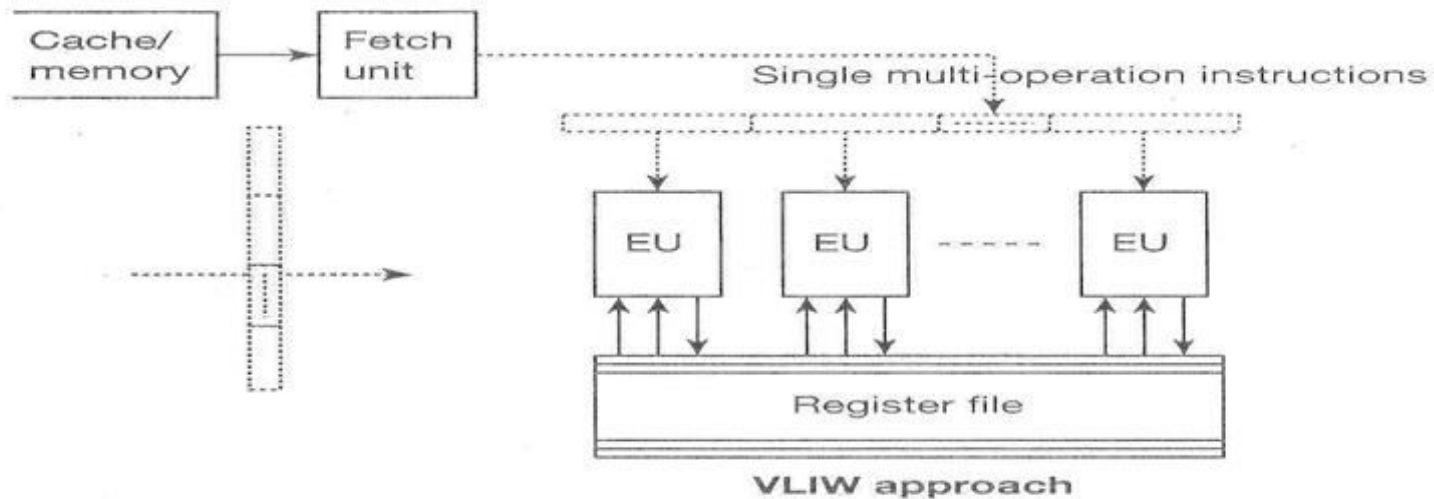


VLIW execution with degree  $m = 3$

# Enabling Technologies for VLIW

- VLIW architectures achieve high performance through the combination of a number of key enabling hardware and software technologies
  - Optimizing scheduler (compilers)
  - Static branch prediction
  - Symbolic memory disambiguation
  - Prediction execution
  - (Software) Speculative Execution
  - Program Compression

# VLIW vs. Superscalar Architecture



Main differences between superscalar processors and VLIW architectures.

# VLIW vs. Superscalar Architecture

## Instruction formulation

### Superscalar:

- Receive conventional instructions conceived for seq. processors.

### VLIW:

- Receive (very) long instruction words, each comprising a field (or opcode) for each execution unit.
- Instruction word length depends (a) number of execution units, and (b) code length to control each unit (such as opcode length, register names, ...).
- 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 lookahead hardware window for instruction fetch.

### VLIW:

- Static scheduling done at compile-time by the compiler.
- **Advantages:**
  - Reduce hardware complexity.
  - Tasks such as decoding, data dependency detection, instruction issue, ..., etc. becoming simple.
  - Potentially higher clock rate.
  - Higher degree of parallelism with global program information.

# VLIW vs. Superscalar Architecture

## Instruction scheduling

### VLIW:

- **Disadvantages**

- Higher complexity of the compiler.
- Compiler optimization needs to consider technology dependent parameters such as latencies and load-use time of cache.

(Question: What happens to the software if the hardware is updated?)

- Non-deterministic problem of cache misses, resulting in worst case assumption for code scheduling.
- In case of un-filled opcodes in a (V)LIW, memory space and instruction bandwidth are wasted.

# Advantages of VLIW

- Parallelism can be exploited at the instruction level
- Hardware is regular and straightforward
- Architecture is compiler friendly
- Exceptions and interrupts are easily managed
- Run-time behavior is highly predictable



# Disadvantages of VLIW

- No object code compatibility between generations
- Program size is large (explicit NOPs)
- Compilers are extremely complex
- Philosophically incompatible with caching techniques
- VLIW memory systems can be very complex
- Parallelism is underutilized for some algorithms

# References

1. PPT on Pipelining from CS303 (3<sup>rd</sup> Semester)
2. *Advanced Computer Architecture – Kai Hwang*
3. *Advanced Computer Architectures – Dezso Sima, Peter Karsuk*
4. Computer Organization – Carl Hamacher
5. Computer Architecture & Organization – John P. Hayes
6. Computer System Architecture – M. Morris Mano
7. Computer Organization & Architecture – T. K. Ghosh
8. Computer Organization & Architecture – Xpress Learning

*Thank You*