
Lecture 10 contd: VLIW and Multiple Instruction Issue Summary

Kunle Olukotun
Gates 302
kunle@ogun.stanford.edu

<http://www-leland.stanford.edu/class/ee282h/>

1

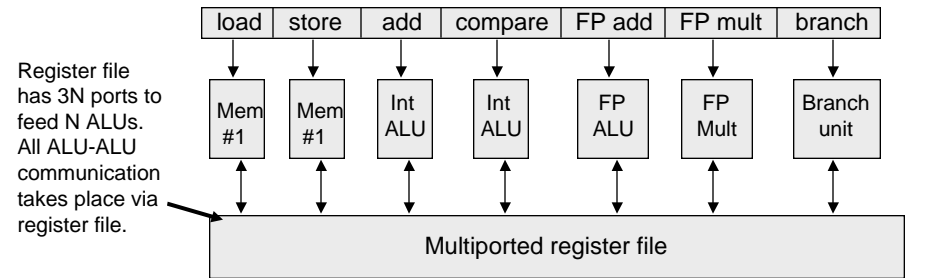
Limitations of Multiple Issue

- Inherent limitations of ILP
 - » Accurate branch prediction is critical
 - » latencies of units require many operations to be scheduled
 - #independent ops = latency * #functional units
- Difficulties in building HW
 - » duplicate FUs to get parallel execution
 - » gates are plentiful, but wires are more expensive in delay and area than gates
 - » increase ports to register file
 - e.g. integer RF : 7 reads and 3 write
 - FP RF: 5 read and 3 write
 - » building a fast multi-ported cache is hard
- Specific limitations of superscalar
 - » decoding and dependency checks for superscalar machine impact clock rate or pipeline depth

2

VLIW: Very Long Instruction Word

- VLIW: tradeoff code density and flexibility for simple decoding
 - » advantage: issue checks (dependencies, resource conflicts) are done at compile time, not execution time
 - » long instruction has room for many operations
 - » all operations in instruction word can execute in parallel
 - » many bits in instruction word
 - 2 integer ops, 2 FP ops, 2 memory refs, 1 branch
 - 16 to 24 bits per field 7*16 or 112 bits to 7*24 or 168 bits wide
 - » Requires compiling technique to schedule across multiple branches



3

Loop Unrolling in VLIW

```

LOOP: LD F0,0(R1)           ; FO = array element
      ADDD F1,F0,F2         ; add scalar in F2
      SD 0(R1),F4           ; store result
      SUBI R1,R1,#8         ; decrement pointer
      BNEZ R1, LOOP        ; branch if R1!=0
  
```

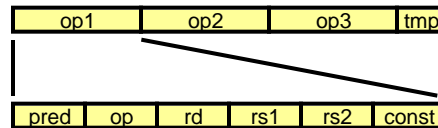
Mem ref 1	Mem ref 2	FP op	FP op	Int op/branch
LD F0,0(R1)	LD F6,-8(R1)			
LD F10,-16(R1)	LD F14,24(R1)			
LD F18,-32(R1)	LD F22,24(R1)	ADDD F4,F0,F2	ADD F8,F6,F2	
LD F26,-48(R1)		ADDD F12,F10,F2	ADDD F16,F14,F2	
		ADDD F20,F18,F2	ADDD F24,F22,F2	
		ADDD F28,F26,F2		
SD 0(R1),F4	SD -8(R1),F8			
SD -16(R1),F12	LD 24(R1),F16			
SD -32(R1),F20	LD 40(R1),F24			
SD 0(R1),F28				
				SUBI R1,R1,#56
				BNEZ R1, LOOP

- » Unrolled 7 times to avoid delays
- » 7 results in 9 clocks or 1.3 clocks per iteration
- » Need more registers in VLIW architecture

4

EPIC

- 128-bit instructions
 - » three 3-address *operations*
 - » a *template* that encodes dependencies
 - » 128 general registers
 - » predication
 - » speculative load



5

VLIW Pros and Cons

- Pros
 - » Very simple hardware
 - no dependency detection
 - simple issue logic
 - just ALUs and register file
 - » Potentially exploits large amounts of ILP
- Cons
 - » Lockstep execution (static schedule)
 - very sensitive to long latency operations (cache misses)
 - » Global register file hard to build
 - » Lots of NOPs
 - poor code 'density'
 - l-cache capacity and bandwidth compromised
 - » Must recompile sources
 - » Implementation visible through ISA

6

Quest for Higher Performance

- Hardware
 - » dynamic branch prediction (local and global correlation)
 - » multiple instruction issue ("superscalar", VLIW)
 - » dynamic scheduling (scoreboard, Tomasulo)
 - » Register renaming (reservation stations, explicit)
 - » speculative execution (reorder buffer)
- Clever compilers can also help
 - » pipeline scheduling
 - » loop unrolling
 - » register renaming
- Big problem - Memory System
 - » Fetching instructions (instruction bandwidth)
 - » Reading and writing data (data bandwidth)