COSC4201
ILP
VLIW, Dynamic scheduling with multiple issue and Predication, and Software techniques

Prof. Mokhtar Aboelaze

Parts of these slides are taken from Notes by Prof. David Patterson (UCB)

VLIW

- Each “instruction” has explicit coding for multiple operations
  - In IA-64, grouping called a “packet”
  - In Transmeta, grouping called a “molecule” (with “atoms” as ops)
- Tradeoff instruction space for simple decoding
  - The long instruction word has room for many operations
  - By definition, all the operations the compiler puts in the long instruction word are independent => execute in parallel
  - E.g., 2 integer operations, 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
  - Need compiling technique that schedules across several branches

VLIW -- Example

Source instruction	Instruction using result	Latency
FP ALU OP	FP ALU OP	3
FP ALU OP	Store double	2
Load double	FP ALU OP	1
Load Double	Store double	0

```
Loop: L.D      F0,0(R1)
      ADD.D    F4,F0,F2
      S.D      0(R1),F4
      DADDUI   R1,R1,#-8
      BNE  R   1,R2,Loop
```

For (I=1000;I>0;I++)

x[I]=x[I]+S;

VLIW -- Example

° Assume that w can schedule 2 memory operations, 2 FP operations, and one integer or branch

Memory reference 1	*Memory reference 2*	*FP operation 1*	*FP op. 2*	*Int. op/ branch*	*Clock*
LD F0,0(R1)	LD F6,-8(R1)				1
LD F10,-16(R1)	LD F14, 24(R1)				2
LD F18,-32(R1)	LD F22,-40(R1)	ADDD F4,F0,F2	ADDD F8,F6,F2	3	
LD F26,-48(R1)		ADDD F12,F10,F2	ADDD F16,F14,F2		4
		ADDD F20,F18,F2	ADDD F24,F22,F2		5
SD 0(R1),F4	SD -8(R1),F8	ADDD F28,F26,F2			6
SD -16(R1),F12	SD -24(R1),F16			DADD R1,R1,#-56	7
SD 24(R1),F20	SD 16(R1),F24				8
SD 8(R1),F28				BNEZ R1,LOOP	9

7 iterations in 9 cycles = 1.29 c/I

VLIW -- Problems

- Increase in code size
  - generating enough operations in a straight-line code fragment requires ambitiously unrolling loops
  - whenever VLIW instructions are not full, unused functional units translate to wasted bits in instruction encoding
- Operated in lock-step; no hazard detection HW
  - a stall in any functional unit pipeline caused entire processor to stall, since all functional units must be kept synchronized
  - Compiler might prediction function units, but caches hard to predict
- Binary code compatibility
  - Pure VLIW => different numbers of functional units and unit latencies require different versions of the code

Advanced Dynamic Scheduling

- Dynamic Scheduling with multiple issue and speculation.
- Two different approaches
  - Issuing the instruction in half a cycle
  - Building the logic to issue 2 instructions simultaneously including detecting dependence
- Must be able to commit more than one instruction at the same time.

Advanced Dynamic Scheduling

```
Loop: LD    R2,0(R1)
      ADD   R2,R2,#1
      SD    R2,0(R1)
      ADD   R1,R1,#8
      BNE   R2,R3,Loop
```


Answer: Without Speculation

Iteration number	Instructions	Issues at clock cycle number	Executes at clock cycle number	Memory access at clock cycle number	Write CDB at clock cycle number	Comment
1	LD R2,0(R1)	1	2	3	4	First issue
1	DADDIU R2,R2,#1	1	5		6	Wait for LW
1	SD R2,0(R1)	2	3	7		Wait for DADDIU
1	DADDIU R1,R1,#4	2	3		4	Execute directly
1	BNE R2,R3,LOOP	3	7			Wait for DADDIU
2	LD R2,0(R1)	4	8	9	10	Wait for BNE
2	DADDIU R2,R2,#1	4	11		12	Wait for LW
2	SD R2,0(R1)	5	9	13		Wait for DADDIU
2	DADDIU R1,R1,#4	5	8		9	Wait for BNE
2	BNE R2,R3,LOOP	6	13			Wait for DADDIU
3	LD R2,0(R1)	7	14	15	16	Wait for BNE
3	DADDIU R2,R2,#1	7	17		18	Wait for LW
3	SD R2,0(R1)	8	15	19		Wait for DADDIU
3	DADDIU R1,R1,#4	8	14		15	Wait for BNE
3	BNZ R2,R3,LOOP	9	19			Wait for DADDIU

Figure 3.33 The time of issue, execution, and writing result for a dual-issue version of our pipeline *without* speculation. Note that the L.D following the BNE cannot start execution earlier, because it must wait until the branch outcome is determined. This type of program, with data-dependent branches that cannot be resolved earlier, shows the strength of speculation. Separate functional units for address calculation, ALU operations, and branch condition evaluation allow multiple instructions to execute in the same cycle.

Answer: 2-way Superscalar Tomasulo With Speculation

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,#4	2	3		4	8	Commit in order
1	BNE R2,R3,LOOP	3	7			8	Wait for DADDIU
2	LD R2,0(R1)	4	5	6	7	9	No execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,#4	5	6		7	11	Commit in order
2	BNE R2,R3,LOOP	6	10		8	11	Wait for DADDIU
3	LD R2,0(R1)	7	8	9	10	12	Earliest possible
3	DADDIU R2,R2,#1	7	11		12	13	Wait for LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,#4	8	9		10	14	Executes earlier
3	BNE R2,R3,LOOP	9	13			14	Wait for DADDIU

Figure 3.34 The time of issue, execution, and writing result for a dual-issue version of our pipeline *with* speculation. Note that the L.D following the BNE can start execution early because it is speculative.

Branches Still Single Issue