

Basic compiler techniques

M. Sonza Reorda

Politecnico di Torino
Dipartimento di Automatica e Informatica

1

Introduction

Compilers can be exploited to reorganize the code for

- Reducing the effects of data hazards
- Attacking branch prediction.

Compiler techniques can be used by themselves or combined with hardware techniques.

2

Scheduling

The compiler can possibly modify the code to exploit parallelism within a basic block.

3

Example

We will consider the effects of the different techniques on a piece of high-level code:

```
for (i=1000; i>0; i=i-1)
    x[i] = x[i] + s;
```

4

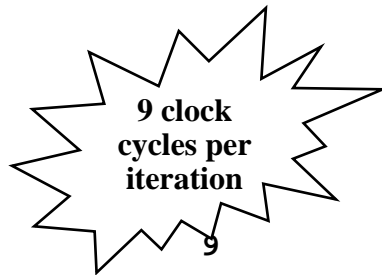
Pipeline behavior

		clock cycle issued
Loop:	L.D F0, 0(R1)	1
	stall	
	ADD.D F4, F0, F2	3
	stall	
	stall	
	S.D F4, 0(R1)	6
	DADDUI R1, R1, #-8	7
	stall	
	BNE R1, R2, Loop	9

7

Pipeline behavior

		clock cycle issued
Loop:	L.D F0, 0(R1)	1
	stall	
	ADD.D F4, F0, F2	3
	stall	
	stall	
	S.D F4, 0(R1)	
	DADDUI R1, R1, #-8	
	stall	
	BNE R1, R2, Loop	9



8

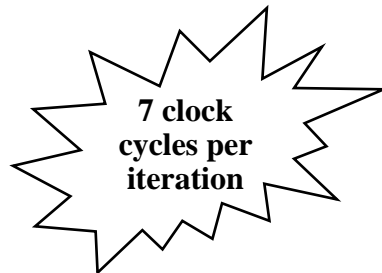
Scheduled assembly code

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

9

Scheduled assembly code

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



10

Observation

Out of the 7 clock cycles required to perform each iteration, only 3 are for computation; the others are for handling the loop.

Further performance optimization can be obtained resorting to loop unrolling.

11

Loop unrolling (basic version)

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

Assumption
the number of iterations
is a multiple of 4.

12

Loop unrolling (basic version)

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

**DADDUI instructions
create dependency
chains on R1 involving
also L.D and S.D
instructions.**

13

Loop unrolling (basic version)

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

**DADDUI instructions
can be removed by
correspondingly
modifying the
addresses for L.D and
S.D instructions.**

14

Loop unrolling (improved version I)

<pre> Loop: L.D F0, 0(R1) ADD.D F4, F0, F2 S.D F4, 0(R1) DADDUI R1, R1, #-8 L.D F0, 0(R1) ADD.D F4, F0, F2 S.D F4, 0(R1) DADDUI R1, R1, #-8 L.D F0, 0(R1) ADD.D F4, F0, F2 S.D F4, 0(R1) DADDUI R1, R1, #-8 L.D F0, 0(R1) ADD.D F4, F0, F2 S.D F4, 0(R1) DADDUI R1, R1, #-8 L.D F0, 0(R1) ADD.D F4, F0, F2 S.D F4, 0(R1) DADDUI R1, R1, #-8 BNE R1, R2, Loop </pre>	<pre> Loop: L.D F0, 0(R1) ADD.D F4, F0, F2 S.D F4, 0(R1) L.D F0, -8(R1) ADD.D F4, F0, F2 S.D F4, -8(R1) L.D F0, -16(R1) ADD.D F4, F0, F2 S.D F4, -16(R1) L.D F0, -24(R1) ADD.D F4, F0, F2 S.D F4, -24(R1) DADDUI R1, R1, #-32 BNE R1, R2, Loop </pre>
--	---

15

Loop unrolling (improved version I)

```

Loop:  L.D      F0, 0(R1)
        ADD.D   F4, F0, F2
        S.D     F4, 0(R1)
        L.D      F0, -8(R1)
        ADD.D   F4, F0, F2
        S.D     F4, -8(R1)
        L.D      F0, -16(R1)
        ADD.D   F4, F0, F2
        S.D     F4, -16(R1)
        L.D      F0, -24(R1)
        ADD.D   F4, F0, F2
        S.D     F4, -24(R1)
        DADDUI  R1, R1, #-32
        BNE     R1, R2, Loop
    
```

Different iterations use the same registers. By using additional registers, different iterations can work in parallel (*register renaming*).

Loop unrolling (improved version II)

Loop:	L.D	F0, 0(R1)	Loop:	L.D	F0, 0(R1)
	ADD.D	F4, F0, F2		ADD.D	F4, F0, F2
	S.D	F4, 0(R1)		S.D	F4, 0(R1)
	L.D	F0, -8(R1)		L.D	F6, -8(R1)
	ADD.D	F4, F0, F2		ADD.D	F8, F6, F2
	S.D	F4, -8(R1)		S.D	F8, -8(R1)
	L.D	F0, -16(R1)		L.D	F10, -16(R1)
	ADD.D	F4, F0, F2		ADD.D	F12, F10, F2
	S.D	F4, -16(R1)		S.D	F12, -16(R1)
	L.D	F0, -24(R1)		L.D	F14, -24(R1)
	ADD.D	F4, F0, F2		ADD.D	F16, F14, F2
	S.D	F4, -24(R1)		S.D	F16, -24(R1)
	DADDUI	R1, R1, #-32		DADDUI	R1, R1, #-32
	BNE	R1, R2, Loop		BNE	R1, R2, Loop

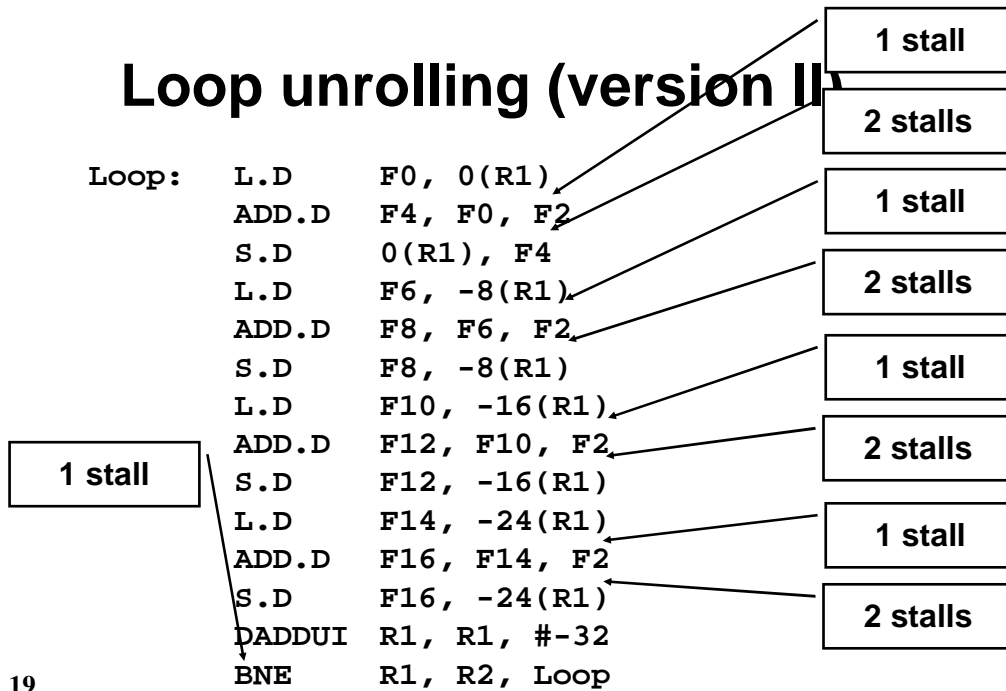
17

Loop unrolling (version II)

Loop:	L.D	F0, 0(R1)
	ADD.D	F4, F0, F2
	S.D	0(R1), F4
	L.D	F6, -8(R1)
	ADD.D	F8, F6, F2
	S.D	F8, -8(R1)
	L.D	F10, -16(R1)
	ADD.D	F12, F10, F2
	S.D	F12, -16(R1)
	L.D	F14, -24(R1)
	ADD.D	F16, F14, F2
	S.D	F16, -24(R1)
	DADDUI	R1, R1, #-32
	BNE	R1, R2, Loop

18

Loop unrolling (version II)



Loop unrolling (version II)

```

Loop:  L.D    F0, 0(R1)
        ADD.D  F4, F0, F2
        S.D    0(R1), F4
        L.D    F6, -8(R1)
        ADD.D  F8, F6, F2
        S.D    F8, -8(R1)
        L.D    F10, -16(R1)
        ADD.D  F12, F10, F2
        S.D    F12, -16(R1)
        L.D    F14, -24(R1)
        ADD.D  F16, F14, F2
        S.D    F16, -24(R1)
        DADDUI R1, R1, #-32
        BNE   R1, R2, Loop
    
```

27 clock cycles, i.e., about 7 clock cycles per iteration

Scheduled loop unrolling

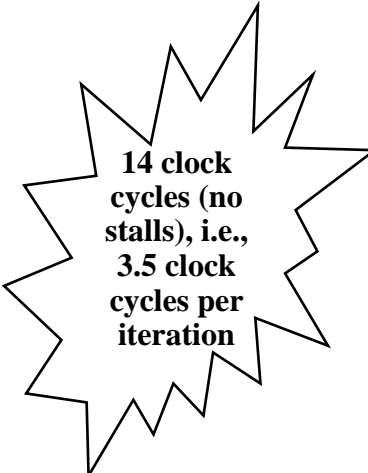
21

```
Loop:  L.D    F0, 0(R1)
       L.D    F6, -8(R1)
       L.D    F10, -16(R1)
       L.D    F14, -24(R1)
       ADD.D  F4, F0, F2
       ADD.D  F8, F6, F2
       ADD.D  F12, F10, F2
       ADD.D  F16, F14, F2
       S.D    F4, 0(R1)
       S.D    F8, -8(R1)
       DADDUI R1, R1, #-32
       S.D    F12, 16(R1)
       S.D    F16, 8(R1)
       BNE   R1, R2, Loop
```

Scheduled loop unrolling

22

```
Loop:  L.D    F0, 0(R1)
       L.D    F6, -8(R1)
       L.D    F10, -16(R1)
       L.D    F14, -24(R1)
       ADD.D  F4, F0, F2
       ADD.D  F8, F6, F2
       ADD.D  F12, F10, F2
       ADD.D  F16, F14, F2
       S.D    F4, 0(R1)
       S.D    F8, -8(R1)
       DADDUI R1, R1, #-32
       S.D    F12, 16(R1)
       S.D    F16, 8(R1)
       BNE   R1, R2, Loop
```



**14 clock
cycles (no
stalls), i.e.,
3.5 clock
cycles per
iteration**

Limitations

- **Decrease in the amount of overhead amortized with each loop unroll**
- **Code size increase**
- **Number of available registers.**