**Computer**

**Architecture**

CH3 Computer Arithmetic (II) Integer Multiplication and Division

Prof. Ren-Shuo Liu

NTHU EE

Fall 2017

Outline • Multiplication

• Shift-add multiplier

• Division

• Shift-subtract divider

2

RISC-V Instructions • Multiplication

• mulh $d, $s1, $s2 # $d=$s1\*$s2 (high 32-bit) • mul $d, $s1, $s2 # $d=$s1\*$s2 (low 32-bit)

• Division

• div $d, $s1, $s2 # $d = $s1 / $s2

• rem $d, $s1, $s2 # $d = $s1 % $s2

3

Multiply Unit, Divide Unit, and Register Files 

From

add/sub unit

… …

R0

R1

R2

R3

…

…

R31

Imm

… …

32 32 mult/div

4

Basic N-Bit Multiplication 

1110 x 0110

1 1 1 0 (14)

x 0 1 1 0 (6)

0 0 0 0

1 1 1 0

1 1 1 0

0 0 0 0

0 1 0 1 0 1 0 0 (84) 2N bits

Multiplicand (被乘數)

Multiplier (乘數)

N iterations; each iteration processes one bit of the multiplier

Product (乘積)

5

Single-Step Multiply 

Clk

Load

A

B

Ready {hi, lo}

1 2 3 4 5 6 7 34 35

14

6

84

receive A & B, A\*B ready

A B Load

32 32

mult

32 32

Clk Ready

6

Single-Step Multiply (Table-Based)

Clk

Load

A

B

Ready {hi, lo}

1 2 3 4 5 6 7 34 35

14

6

84

receive A & B, A\*B ready

A B

32 32

A B

64

264-entry

look-up table

32 32

• Theoretically doable

• Rreasonable for small N (e.g., 4) • **Impractical** for large N (e.g., 32)

7

Single-Step Multiply (Adder-Based)

1 2 3 4 5 6 7 34 35 Clk

Load

A B

32 32

A B

…

A

14

<< 0

B[0] B[1]

<< 1

<< 2

B[2] B[3]

<< 3

<< 30

B[30] B[31]

<< 31

B

Ready {hi, lo}

6

84

+

+

+ … +

…

+

receive A & B, A\*B ready

• Total (16+8+4+2+1)=31 adders

8

N-Step Multiply 

Clk

Load

A

B

Ready {hi, lo}

1 2 3 4 5 6 7 34 35

14

6

84

A B Load

32 32

mult

Clk Ready

9

N-Step Multiply 

Clk

Load

A

B

Ready {hi, lo}

1 2 3 4 5 6 7 34 35

14

6

84

receive A & B A\*B ready

A B

32 32 A B

+

Hi Lo

state

10

N-Step Multiply 

1 2 3 4 5 6 7 34 35

1 1 1 0

Clk

x 0 1 1 0

P stands for {Hi, Lo}

Load

A

B

Ready {hi, lo}

14

6

84

32 steps

receive A & B A\*B ready

P **=** P

P **=** P **+** A<<1) P **=** (P **+** A<<2) P **=** P

0 0 0 0

1 1 1 0

1 1 1 0

0 0 0 0

0 1 0 1 0 1 0 0

11

Basic Multiply Unit **A B**

****Load

www

……

sh

…… ……

rst

64

64

+

**Hi Lo**

…

31

w

1 0

6 1 +

… …

rst

Control Ready

12

t = 0.99



… …

1 2 3 4 5 6 7 8 9

www

…… …… ……

64

sh

31

…

rst

1 0

6 1

14 6

~~64~~

+

w

+

**~~Hi~~ Lo** Control

~~…~~ … 84

rst

…

13

t = 1.2 

14… …6

1 2 3 4 5 6 7 8 9

www

…… …… ……

64

sh

31

…

rst

1 0

6 1

14 6

~~64~~

+

w

+

**~~Hi~~ Lo** Control

~~…~~ … 84

rst

14

t = 1.99 

14… …6

1 2 3 4 5 6 7 8 9

www

…… …… ……

64

sh

31

…

rst

1 0

6 1

14 6

~~64~~

+

w

+

**~~Hi~~ Lo** Control

… … 84

rst

15

t = 2.2

• Received A & B. Reset Hi & Lo. 

• Start preparing next {Hi, Lo} based on B[0]=0.

… …

1 2 3 4 5 6 7 8 9

www

…… …….. …… ……

14 6 0

64

sh

31

…

rst

1 0

6 1

14 6

~~64~~

+

w

+

0 Control

…….

84

rst

16

t = 2.99

2 3 4 5 6 7 8 9

• Prepared next {Hi, Lo} based on B[0]=0. • Start transitioning to the next {Hi, Lo}

… …

www

…… …….. …… ……

14

sh

6 0

rst

4 6

~~64~~

64

+

0

w

31

…

1 0

6 1 + 

…….

84

rstControl

17

t = 3.2

2 3 4 5 6 7 8 9

• Hi & Lo changed. 

• Start preparing next {Hi, Lo} based on B[1]=1.… …

www

…… …….. …… ……

14x2

sh

6 1

rst

4

~~64~~

64

+

w

31

…

1 0

6 1 + 

0 Control

84

…….

rst

18

t = 3.99

3 4 5 6 7 8 9

• Prepared next {Hi, Lo} based on B[1]=1. • Start transitioning to the next {Hi, Lo}

… …

www

…… …….. …… ……

14x2

sh

6 1

rst

64

64

+

w

31

…

1 0

6 1 + 

0 Control

84

…….

rst

19

t = 4.2

3 4 5 6 7 8 9

• Hi & Lo changed. 

• Start preparing next {Hi, Lo} based on B[2]=1.… …

www

…… …….. …… ……

14x4

sh

6 2

rst

64

64

+

w

31

…

1 0

6 1 + 

14x2 Control

84

…….

rst

20

t = 4.99

4 5 6 7 8 9

• Prepared next {Hi, Lo} based on B[2]=1. • Start transitioning to the next {Hi, Lo}

… …

www

…… …….. …… ……

14x4

sh

6 2

rst

64

64

+

w

31

…

1 0

6 1 + 

14x2 Control

84

…….

rst

21

t = 5.2

4 5 6 7 8 9

• Hi & Lo changed. 

• Start preparing next {Hi, Lo} based on B[3]=0.… …

www

…… …….. …… ……

14x8

sh

6 3

rst

64

64

+

w

31

1 0

6 1 + 

14x2 + 14x4 Control

84

…….

rst …

22

t = 5.99

4 5 6 7 8 9

• Prepared next {Hi, Lo} based on B[3]=0. • Start transitioning to the next {Hi, Lo}

… …

www

…… …….. …… ……

14x8

sh

6 3

rst

64

64

+

w

31

1 0

6 1 + 

14x2 + 14x4 Control

84

…….

rst

23

t = 6.2

5 6 34 9

• Hi & Lo changed. 

• Start preparing next {Hi, Lo} based on B[4]=0.… …

www

…… …….. …… ……

14x16

sh

6 4

rst

64

64

+

w

31

1 0

6 1 + 

14x2 + 14x4 Control

…….

rst

24

t = 33.99 34 35

• Prepared next {Hi, Lo} based on B[31]=0. • Start transitioning to the next {Hi, Lo}

… …

www

…… …….. …… ……

14 << 31

sh

6 31

rst

64

64

+

w

31

1 0

6 1 + 

14x2 + 14x4 Control

84

…….

rst

25

t = 34.2

34 35

• Hi & Lo are done. 

• Raise the Ready signal.

… …

www

…… …….. …… ……

14 << 31

sh

6 31

rst

64

64

+

w

31

1 0

6 1 + 

14x4 + 14x2 Control

84

…….

rst 

26

Registers with Reset and Write  

64

W

Rst

Bit 63 Bit 1 Bit 0

0

0

00~~D~~ Q

0~~D~~ Q

Clk

1

1

0

~~D~~ Q

64

…

1

1

0

1

0

0 1

27

Shift Registers 

0

**1** 0 0

~~D~~ Q

Clk

~~D~~ Q ~~D Q~~

0 **1** 0

0

~~D~~ Q ~~D Q~~

Clk

~~D~~ Q

28

Registers with Write and Left Shift 

Bit 63 Bit 1 Bit 064

…

W

Shift

00

~~D~~ Q

0

00

~~D~~ Q

00

~~D~~ Q

01 … 01 01

Clk

10

64

10

10

29

The Control Unit

**A B**

****Load

www

……

sh

…… ……

rst

64

+

…

31

1 0

6 1 +

64

…….

w

rst

b L nControl

Ready

30

The Control Unit 

n

bL

…

w0 = f0(n, L, b)w1 = f1(n, L, b) w2 = f2(n, L, b) rst0

rst1

sh

ready

31

The Control Unit 

rst0 = L;

rst1 = L;

w0 = ( n < 33 );

**A B**

Load

w1 = L;

w2 = (b && ( n < 32 ));

……

w1w1w0 …… ……

sh = ( n < 32 ); Ready = (n == 32 );

64

sh

31

…

rst0

1 0

6 1

64

+

…….

w2 rst1 

b L n 

Control



Ready

+

32

Improved Multiply Unit (V1)

**A B**

• We can use shift registers

here to replace the 32-to

Load

1 multiplexer

www

……

left sh

…… ……

rst

64

64

+

w

31

…

1 0

6 1 +

…….

rst

Control

**P** Overflow Ready

33

Improved Multiply Unit (V1) **A B**

****Load

w w w

64

64

+

**Hi Lo**… …

sh

w

rstControl Ready

sh

……

6 1 +

rst

34

Improved Multiply Unit (V2) **A B**

• 64-bit adder is

Load

wasteful

• Each iteration only

w w w

……

touches the 32-bit Hi

64

+

left sh

r sh 

register

• Solution

rst

6 1

+

64

w

• Right shifting {Hi, Lo} instead of left

shifting the

… …

rstControl

multiplicand

**P** Overflow Ready

35

Improved Multiply Unit (V2) **A B**

****Load

w w w

32

32

+

**Hi Lo**

sh w

rst

sh

Control

Ready

……

6 1 +

rst

36

Improved Multiply Unit (V3)

**A B**

• Lo is not in use in the

beginning

Load

w w w

32 +

left sh

sh

……

6 1 +

rst

32

**Hi Lo**

sh w

rst

🡪 sharing registers between multiplicand Control

and Lo

**P** Overflow Ready

37

Improved Multiply Unit (V3) 

**A B**

Load

w w

32

32 +

sh w

……

6 1 +

rst

**Hi Lo** Controlrst

Ready

38

Improved Multiply Unit (V3) • A = 1110 = 14ten, B = 0110 = 6ten

| 0 | **Hi**  0 0 0 0 | **Lo (B)**  0 1 1 0 | **Lo[0]** | **Next Action Decided by Control** |
| --- | --- | --- | --- | --- |
| 0 | Right shift {Hi, Lo} |
| 1 | 0 0 0 0 | 0 0 1 1 |  |  |
| 1 | HiD = (HiQ + A); Right shift {Hi, Lo} |
| 2 | 0 1 1 1 | 0 0 0 1 |  |  |
| 1 | HiD = (HiQ + A); Right shift {Hi, Lo} |
| 3 | 1 0 1 0 | 1 0 0 0 |  |  |
| 0 | Right shift {Hi, Lo} |
| 4 | 0 1 0 1 | 0 1 0 0 |  |  |
|  |  |

{Hi, Lo} = 0101\_0100 = 84ten

39

Simultaneous Add and Right Shift

0

5

{0, X[3:0]}

X[3:0] Y[3:0]

+

{Cout, S[3:0]}

Action sel.

D7 D6 D5 D4 D3 D2 D1 D0 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

40

Improved Multiply Unit (V3) 

**A B**

Load

w w

32

32 +

sh w

……

6 1 +

rst

**Hi Lo** Controlrst

Ready

41

Signed Multiplication 

1110 x 0110 = - ((-1110) x 0110) = - (0010 x 0110)

0 0 1 0 (2)

x 0 1 1 0 (6)

0 0 0 0

0 0 1 0

0 0 1 0

0 0 0 0

0 0 0 1 1 0 0 (12) Negate

1 1 1 0 1 0 0 (-12)

Signed multiplication can be done by using the unsigned procedure

42

MIPS Division Instructions • DIV $d, $s, $t

• REM $d, $s, $t

43

Division

0 0 0 0 1 0 0 1 

Quotient (商) Divisor (除數)

0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0

0 0 0 0 1 0 0 0

0 0 0 0 1 0 0 0

0 0 0 0 1 0 0 0

0 0 0 0 1 0 0 0

0 0 0 0 1 0 1 0

Dividend (被除數)

These are trial

subtractions, which result in negative

dividend.

N subtractions

0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0

Remainder (餘數)

Basic Division Unit

**A B**

0

B A BBB 

Load

w w

B

64

64

+

sh

B negate Cout

1

sh 

sh 

in

……

6 1 +

rst

rst A

w

Control

**A/B**

**A%B** Ready

45

B A

Improved Division Unit (V1) **A B**

****B Load

A 

A

ww

32

32

B

32

B negate + 1 Cout

whi

Qsh sh 

in

……

6 1 +

rst

rst

A

wlo sh

Control

**A/B**

**A%B** Ready 46

Improved Division Unit (V2) **A B**

****Load

ww

32

32

B

32

B negate + 1 Cout

whi

wlo

……

6 1 +

rst

rst Control

A

sh 

sh in

**Lo = A/B**

**Hi = A%B** Ready 47

Improved Division Unit (V2) • A = 01001010 = 74ten, B = 00001000 = 8ten

| 0 | **Hi (R)**  0 0 0 0 0 0 0 0 | **Lo (A, Q)**  0 1 0 0 1 0 1 0 | **Hi < B?** | **Todo** |
| --- | --- | --- | --- | --- |
| 1 | Shift **0** into {Hi, Lo} |
| 1 | 0 0 0 0 0 0 0 0 | 1 0 0 1 0 1 0 **0** |  |  |
| 1 | Shift **0** into {Hi, Lo} |
| 2 | 0 0 0 0 0 0 0 1 | 0 0 1 0 1 0 **0 0** |  |  |
| 1 | Shift **0** into {Hi, Lo} |
| 3 | 0 0 0 0 0 0 1 0 | 0 1 0 1 0 **0 0 0** |  |  |
| 1 | Shift **0** into {Hi, Lo} |
| 4 | 0 0 0 0 0 1 0 0 | 1 0 1 0 **0 0 0 0** |  |  |
| 1 | Shift **0** into {Hi, Lo} |
| 5 | 0 0 0 0 1 0 0 1 | 0 1 0 **0 0 0 0 0** |  |  |
| 0 | Hi = (Hi-B); Shift **1** into {Hi, Lo} |
| 6 | 0 0 0 0 0 0 1 0 | 1 0 **0 0 0 0 0 1** |  |  |
| 1 | Shift **0** into {Hi, Lo} |
| 7 | 0 0 0 0 0 1 0 1 | 0 **0 0 0 0 0 1 0** |  |  |
| 1 | Shift **0** into {Hi, Lo} |
| 8 | 0 0 0 0 1 0 1 0 | **0 0 0 0 0 1 0 0** |  |  |
| 0 | Hi = (Hi-B); Shift **1** into {Lo} |
| 9 | 0 0 0 0 0 0 1 0 | **0 0 0 0 1 0 0 1** |  |  |
|  |  |

Remainder (2) Quotient (9)

48

Signed Division • Rule

• The remainder (餘數) should have the same signs as that of dividend (被除數)

• A ÷ B

• |Q| = |A| / |B|; sign(Q) = sign(A) XOR sign(B)

• |R| = |A| % |B|; sign(R) = sign(A)

• Examples

• 7 ÷ 2 = 3 … 1

• (-7) ÷ 2 = (-3) … (-1)

• 7 ÷ (-2) = (-3) … 1

• (-7) ÷ (-2) = 3 … (-1)

49

Summary • Bit-wise logical 

• and, andi 

• or, ori 

• xor, xori 

• nor

• Arithmetic 

• add, addu, addi, addiu 

• sub, subu

• mulh, mul 

• div, rem 

• Comparisons

• slt, slti, sltu, sltiu

50

Note • My slides present slightly different from the textbook

• Textbook plots the flow charts of performing multiplication and division

• My slides emphasize logic circuits

• In the textbook, the dividend is subtracted by the divisor before the subtraction result is checked

• In my slides, the subtraction result is checked before the result is written into the dividend register

• Please also read the textbook, too

51

Outline • Overview

• Integer operations

• Bit-wise logical operations

• Additions, subtractions

• Comparisons

• Multiplications

• Divisions

• Floating point operations

• Additions, subtractions

• Multiplications

52