Lecture 4
Multiplication and division

Computing platforms

Novosibirsk State University
University of Hertfordshire

D. Irtegov, A.Shafarenko

2018

Multiplication by adding

- A*B

```
P=0
While B>0 do
    P+=A
    B--
Wend
```

- For 8-bit values, 256 additions in worst case
- For 64-bit values on modern CPU, won’t finish in your lifetime

Let’s consider special cases

- A*2 = A+A = lshift(A,1)
- A*2^N = while N-->0 do P+=A wend = lshift(A,N)
- A*(2^N+2^M)=A*2^N+A*2^M
- If we represent arbitrary number as sum of 2^N…

Algorithm of multiplication

- Any number has the binary representation
- $B=Sum(b[N]*2^N)$, where b[N] - Nth bit of binary representation
- $P=A*B=Sum(A*b[N]*2^N)$
- So, the algorithm

```
N=0
P=0
While N<bits(B) do
    P+=A*b[N]
    A=lshift(A,1)
Wend
```

Let’s try to visualize it

Note that 4-bit*4bit yields 8-bit result

$$
\begin{array}{r cccccccc}
 & & & & & 1 & 1 & 0 & 1 \\
\times & & & & & 1 & 1 & 1 & 0 \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
\hline
 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

Looks familiar?

$$
\begin{array}{c cccccccc}
 & & & & & 1 & 1 & 0 & 1 \\
\times & & & & & 1 & 1 & 1 & 0 \\
\hline
 & & & & & 0 & 0 & 0 & 0 \\
 & & & & 1 & 1 & 0 & 1 & \\
 & & & 1 & 1 & 0 & 1 & & \\
 & & 1 & 1 & 0 & 1 & & & \\
\hline
 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

How to implement this in CdM-8?

- b[N] can be calculated as series of right shifts
- Shr instruction shifts the register and moves lowest bit to C
- We do not need to count to 8
- The loop can stop when reg==0 (Z flag is set)
- But how to calculate 16-bit P and 16-bit $A*2^N$?
- They need 2 registers each, and we have only four registers.

Let’s go in other direction

```
N=7
P=0
While True do
   P+=A*b[N]
   if N==0 break
   P=rshift(P,1)
   N--
Wend
```

- Now we need a register to store N
- Or we can unroll the loop (there are only 8 iterations after all)

Demonstration in CocoIDE

- http://ccfit.nsu.ru/~fat/Platforms/mult.asm
- 8-bit unsigned multiplication witn 16-bit results using only registers (no memory access)

What about signed multiplication?

$$
\begin{array}{r}
1\,1\,0\,1 \\
\times \qquad\quad 1\,1\,1\,0 \\
\hline
0\,0\,0\,0 \\
1\,1\,0\,1 \\
1\,1\,0\,1 \\
1\,1\,0\,1 \\
\hline
1\,0\,1\,1\,0\,1\,1\,0
\end{array}
$$

If we treat 1101 and 1110 as two-complement signed numbers, the result is wrong.
You do not even need to convert to decimal.
The operands are both negative, but the result is positive!

Proper way of two-complement signed multiplication

Sign-extend both numbers *before* the multiplication

Actually, this is a disadvantage of two-complement presentation

With sign-magnitude, you just multiply unsigned and xor sign bits

$$
\begin{array}{r}
1\,1\,1\,1\,1\,1\,0\,1 \\
\times\ 1\,1\,1\,1\,1\,1\,1\,0 \\
\hline
0\,0\,0\,0\,0\,0\,0\,0 \\
1\,1\,1\,1\,1\,0\,1\,0 \\
1\,1\,1\,1\,0\,1\,0\,0 \\
1\,1\,1\,0\,1\,0\,0\,0 \\
1\,1\,0\,1\,0\,0\,0\,0 \\
1\,0\,1\,0\,0\,0\,0\,0 \\
0\,1\,0\,0\,0\,0\,0\,0 \\
1\,0\,0\,0\,0\,0\,0\,0 \\
\hline
0\,0\,0\,0\,0\,1\,1\,0
\end{array}
$$

Division

- the dividend is the number to be divided
- the divisor is the number the dividend is divided by
- the quotient is the main result of division,
- a remainder, which is the quantity left over, i.e. the difference between the dividend and product of the quotient and the divisor.

Exact definition of quotient

- a quotient, which is the whole number of times the divisor ‘goes into’ the dividend.
- In other words, the quotient is the maximum integer that if multiplied by the divisor gives the result not exceeding the dividend.

Let’s try to divide 11(dec) to 3(dec)

- 11÷3=3 rem 2
- 11(dec)=1011
- 3=0011

STAGE 1:

$$\begin{array}{r@{\,}l}
 & 1 \\ \cline{2-2}
11 & \big)\,1\,0\,1\,1 \\
 & -\;1\,1 \\ \cline{2-2}
 & 1\,0
\end{array}$$

STAGE 2:

$$\begin{array}{r@{\,}l}
 & 1 \\ \cline{2-2}
11 & \big)\,1\,0\,1\,1 \\
 & -\;1\,1 \\ \cline{2-2}
 & 1\,0\,1
\end{array}$$

STAGE 3:

$$\begin{array}{r@{\,}l}
 & 1\,1 \\ \cline{2-2}
11 & \big)\,1\,0\,1\,1 \\
 & -\;1\,1 \\ \cline{2-2}
 & 1\,0\,1 \\
 & -\;1\,1 \\ \cline{2-2}
 & 1\,0
\end{array}$$