Computer Mathematics

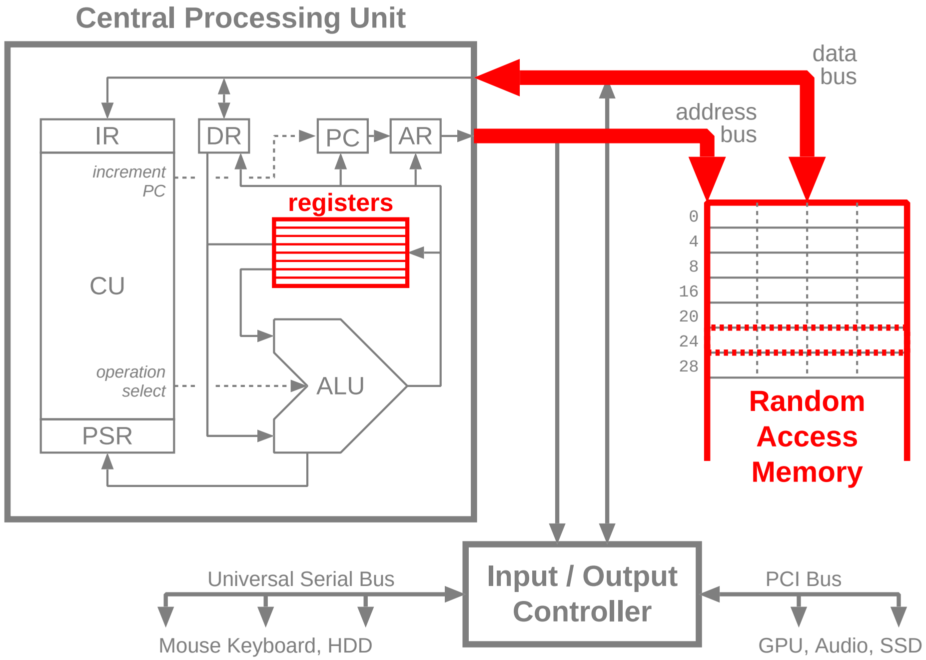
Week 3
Unsigned integer arithmetic

京都先端科学大学
KYOTO UNIVERSITY of ADVANCED SCIENCE
Department of Mechanical and Electrical System Engineering

last week

positional number representations

- digit positions have weights (significance)
- related by the radix (base)
- weight $\times$ digit $=$ value
- number is the sum of the values



converting an integer to radix r

- divide by r, remainder is digit, repeat until 0
  (digits generated from right to left)

converting a fraction to radix r

- multiply by r, integer part is digit, discard integer part, repeat until 0
  (digits generated from left to right)

useful bases: binary ($r = 2$), octal ($r = 8$), decimal ($r = 10$), hexadecimal ($r = 16$)

this week

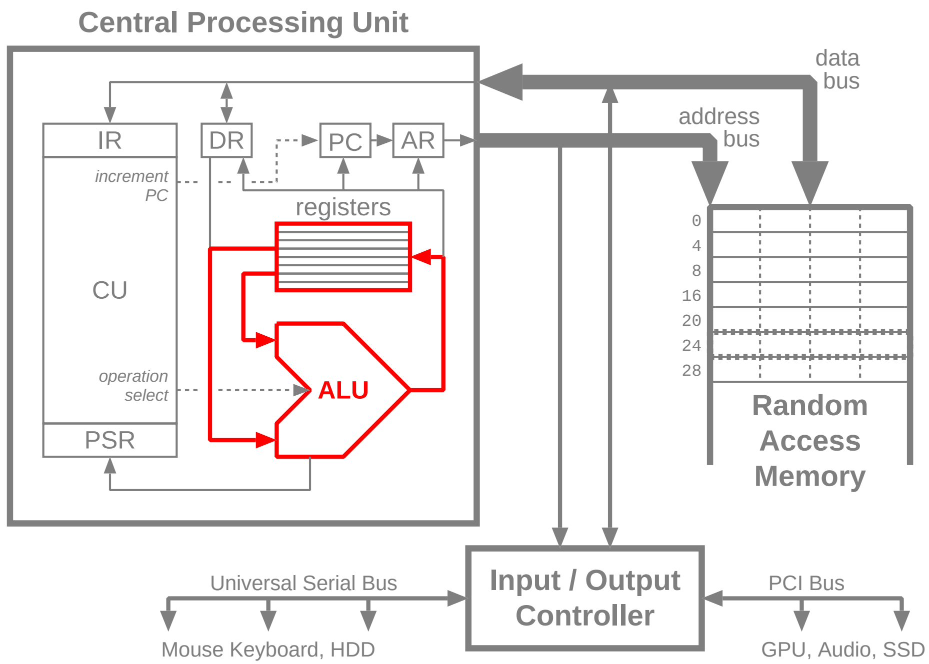
range of representable numeric values

unsigned arithmetic

- terminology
- basic mathematical operations
  - $+ \quad - \quad \times \quad \div$
  - in decimal
  - in binary

integer overflow

- conditions and detection



arithmetic

a *unary operator* applies a mathematical *operation* to one *operand*

$$- \ z$$

$$\uparrow \quad \uparrow$$

operator operand

a *binary operator* applies a mathematical operation to two operands

$$x \ + \ y$$

$$\uparrow \quad \uparrow \quad \uparrow$$

operand operator operand

unsigned number range

for an unsigned, n-digit, radix-r number

- the minimum value is when all digits are 0
- the maximum value is when all digits are $r-1$

e.g., with four-digit numbers, the maximum representable values are

base	*number*	*value*	
decimal $(r=10)$	9999	9999_{10}	$= 10000 - 1 = 10^4 - 1$
binary $(r=2)$	1111	15_{10}	$= 16 - 1 = 2^4 - 1$

when performing binary arithmetic on n-digit numbers

- minimum allowed operand or result: 0
- maximum allowed operand or result: $2^n - 1$

operands will always be within these limits

results outside the range $[0 \ldots 2^n - 1]$ cannot be represented; they have *overflowed*

modular arithmetic

adding one digit to another produces a single-digit result; in decimal:

$$3 + 4 = 7$$
$$7 + 8 = 5 \quad \textit{(carry one)}$$

when the result exceeds 9, it '*wraps* around' back to 0 (and generates a *carry*)

subtracting one digit from another produces a single-digit result; in decimal:

$$4 - 3 = 1$$
$$3 - 4 = 9 \quad \textit{(borrow one)}$$

when the result is less than 0, it 'wraps around' back to 9 (and generates a *borrow*)

for radix r numbers, this is called 'modulo-r' arithmetic

- e.g., 'modulo-10' for decimal, and 'modulo-2' for binary

(carries and borrows are always either 0 or 1)

unsigned addition

adding two multi-digit numbers is just adding individual digits, including carries

$$\begin{array}{r} 1\,7 \leftarrow \textit{augend} \\ +\,2\,8 \leftarrow \textit{addend} \\ \textit{carry out} \text{ from '}7+8\text{'} = \textit{carry in} \text{ to '}1+2\text{'} \longrightarrow 1 \\ \hline 4\,5 \leftarrow \textit{sum} \end{array}$$

in binary, using 4-bit words:

$$\begin{array}{r} 0\,1\,1\,0 \\ +\,0\,1\,1\,1 \\ \text{carry out} \longleftarrow 0\;1\,1\,0\,0 \longleftarrow \text{carry in} \\ \hline 1\,1\,0\,1 \end{array}$$

representable range depends on number of bits in a word

with 4 bits:

- minimum representable value $0000_2 = 0$, maximum value $1111_2 = 15$, but. . .
- maximum sum $= 15 + 15 = 30$ (11110_2)
- if *carry out* $= 1$ then the sum was > 15, and the addition *overflowed*
  - the result cannot be represented in the number of bits available

unsigned subtraction

subtracting two multi-digit numbers is just subtracting individual digits, including borrows

$$
\begin{array}{rl}
4\,7 & \leftarrow \textit{minuend} \\
-\;1\,8 & \leftarrow \textit{subtrahend} \\
\textit{borrow out} \text{ from } `7-8\text{'} = \textit{borrow in} \text{ to } `4+1\text{'} \longrightarrow 1 & \\
\hline
2\,9 & \leftarrow \textit{difference}
\end{array}
$$

in binary, using 4-bit words:

$$
\begin{array}{rcl}
 & 1\,1\,0\,1 & \\
 & -\;0\,1\,1\,1 & \\
\text{borrow out} \longleftarrow 0 & 1\,1\,0\,0 & \longleftarrow \text{borrow in} \\
\hline
 & 0\,1\,1\,0 &
\end{array}
$$

overflow occurs if the answer should be negative

- if *borrow out* $= 1$ then the difference was < 0, and the subtraction overflowed
  - the result cannot be represented as an unsigned number

unsigned multiplication

to simplify multiplication we

- split one of the operands into individual digit values
- multiply the other operand by each individual digit value
- sum all the resulting *partial products*

$$
\begin{array}{rrrl}
 & 2\,3\,4 & & \leftarrow \textit{multiplicand} \\
\times & 2\,1\,1 & & \leftarrow \textit{multiplier} \\
\hline
 & 2\,3\,4 & (1 \times 234) & \\
 & 2\,3\,4\,0 & (10 \times 234) & \leftarrow \textit{partial products} \\
+ & 4\,6\,8\,0\,0 & (200 \times 234) & \\
 & {\color{red}1} & & \\
\hline
 & 4\,9\,3\,7\,4 & & \leftarrow \textit{product} \\
\hline
\end{array}
$$

unsigned multiplication

this is very easy in binary

- multiplying by 0 or 1 is trivial

$$
\begin{array}{rrl}
 & 1\,0\,1\,1 & \\
\times & 1\,1\,1\,0 & \\
\hline
 & 0\,0\,0\,0 & (0 \times 1011) \\
 & 1\,0\,1\,1 & (10 \times 1011) \\
 & 1\,0\,1\,1 & (100 \times 1011) \\
+ & 1\,0\,1\,1 & (1000 \times 1011) \\
 & \color{red}{1\,1\,1\,1\,1} & \\
\hline
 & 1\,0\,0\,1\,1\,0\,1\,0 & \\
\hline
\end{array}
$$

note that the leftmost carry is also part of the result

multiplying two n-bit numbers creates a $2n$-bit result

- e.g., $1111_2 \times 1111_2 = \color{red}{1110}\,0001_2$

overflow occurs if the answer is $\geq 2^n$

- i.e., if any of the most-significant n bits are non-zero

unsigned division

division is performed by repeated subtraction

$$
\begin{array}{rr|ccccl}
 & & 0 & 3 & 2 & & \leftarrow \textit{quotient} \\
\cline{3-5}
\textit{divisor} \rightarrow & 7 & 2 & 3 & 0 & & \leftarrow \textit{dividend} \\
 & - & 2 & 1 & \downarrow & & \leftarrow 7 \times 3 \\
\cline{3-4}
 & & & 2 & 0 & & \\
 & & - & 1 & 4 & & \leftarrow 7 \times 2 \\
\cline{4-5}
 & & & & 6 & & \leftarrow \textit{remainder}
\end{array}
$$

$$230 \div 7 = 32\,\mathrm{r}\,6$$

overflow cannot occur

division by zero is disallowed

- the answer is *indeterminate*

unsigned division

this is very easy in binary

- at each step, the divisor can be subtracted exactly zero or one times

$$
\begin{array}{r|cccccc}
 & & & & 1 & 1 & 0 \\ \cline{2-7}
1001 & 1 & 1 & 1 & 0 & 0 & 0 \\
\multicolumn{1}{r}{} & 1 & 0 & 0 & 1 & \color{red}{\downarrow} & \\ \cline{2-5}
\multicolumn{1}{r}{} & & 1 & 0 & 1 & 0 & \\
\multicolumn{1}{r}{} & & 1 & 0 & 0 & 1 & \color{red}{\downarrow} \\ \cline{3-6}
\multicolumn{1}{r}{} & & & & & 1 & 0
\end{array}
$$

$$111000_2 \div 1001_2 = 110_2 \,\mathrm{r}\, 10_2$$

$$(56_{10} \div 9_{10} = 6_{10} \,\mathrm{r}\, 2_{10})$$

summary

binary addition

- performed like decimal addition, but using only two digits
- like decimal, carry into next column is always 0 or 1
- like decimal, carry out of most significant position indicates overflow
  - the result is too large

binary subtraction

- performed like decimal subtraction, but using only two digits
- like decimal, borrow from next column is always 0 or 1
- like decimal, borrow from most significant position indicates overflow
  - the result is negative

summary

binary multiplication

- performed like decimal multiplication, but using only two digits
- like decimal, two n-digit numbers can produce a $2n$-digit result
- like decimal, any non-zero digit in the leftmost n digits indicates overflow
  - the result is too large

binary division

- performed like decimal division, but using only two digits
- the quotient can never overflow (it is always smaller than the dividend)
- the remainder can never overflow (it is always in the range $[0 \ldots divisor - 1]$)
- like decimal, division by zero is an error
  - the result is indeterminate

next week

negative numbers

- various representations
- advantages, disadvantages

signed binary arithmetic

- negation
- subtraction made easy
- signed overflow

homework

next week's class will combine knowledge from this week and last week

review the slides for weeks 2 and 3

review the handouts for weeks 2, 3 and 4:

- Chapter 2: Positional number systems
- Chapter 3: Binary arithmetic

(the material in the handouts will make more and more sense after each week's class)

from the handouts, try to understand

- one's complement negative representation
- two's complement negative representation
- signed addition and subtraction

before coming to next week's class

glossary

binary operator — an operator that applies an operation to two operands.

borrow — a 1 that is transferred from the next more significant position (one position to the left). In radix-r modular artithmetic, a borrow is needed when the difference d of two digits is negative and cannot be stored in the result. Adding r to d yields a positive, single-digit result that can be stored in the result. To compensate for this, r needs to be subtracted from the overal difference. Since the column to the left has a weight r times larger, subtracting an extra 1 from the result digit in that column provides this compensation.

borrow in — the borrow generated in the next less significant position (one position to the right).

borrow out — the borrow that will be propagated to the next more significant position (one position to the left).

carry — a 1 that is transferred to the next more significant position (one position to the left). In radix-r modular artithmetic, a carry is generated when the sum s of two digits is $\geq r$ and cannot be stored in the result. Subtracting r from s yields a single-digit result that can be stored in the result. To compensate for this, r needs to be added to the overal sum. Since the column to the left has a weight r times larger, adding an extra 1 to the result digit in that column provides this compensation.

carry in — the carry generated in the next less significant position (one position to the right).

carry out — the carry that will be propagated to the next more significant position (one position to the left).

glossary

indeterminate — a result that can have many possible values.

modular arithmetic — a system of integer arithmetic in which the range of possible values is restricted by a modulus m. When a value is increased to m, it instead wraps around to 0. When a value is decreased to -1, it instead wraps around to $m - 1$. The behaviour is equivalent to dividing all arithmetic results by m and then using the remainder. Such arithmetic is said to be performed 'modulo m'.

operand — an input to a mathematical operation.

operation — a mathematical function such as addition, subtraction, multiplication, division, negation.

operator — a symbol in mathematics representing an operation to be applied to one or more operands.

overflow — a condition that occurs when the result of an operation cannot be represented in the available number of bits.

partial product — the result of multiplying the multiplicand by one digit of the multiplier. The product of multiplicand and multiplier is the sum of the partial products.

saturate —

unary operator — an operator that applies an operation to one operand.

wrap around — in modulo m arithmetic, the return to 0 that occurs when an increasing quantity passes $m - 1$ (or the return to $m - 1$ that occurs when a decresing quantity passes 0).