CSE 20 Discussion: Week 4

Number representations, circuits, propositional logic

Logistics: Q&A

- Sign up for a time slot that works for your HW2 teammates. Spots are filling up quickly!
- Print your submitted HW1 and HW2 PDFs exactly as submitted and bring the physical copies to Q&A day.
  - If your team changed between HW1 and HW2, feel free to bring both versions or use whichever version all teammates are comfortable with.
- Review the released solutions for both homework assignments. The HW2 solution will be released shortly after the deadline.
- Arrive 10 minutes before your scheduled Q&A time.
- For weekend access, use the PIN code and do not share it with anyone outside the class.

Review (Week 2): base expansions

Definition: For b an integer greater than 1 and n a positive integer, the **base b expansion of n** is

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are (symbols for) nonnegative integers less than b, $a_{k-1} \neq 0$, and

$$n = \sum_{i=0}^{k-1} a_i b^i$$

Notice: *The base b expansion of a positive integer n is a string over the alphabet $\{x \in \mathbb{N} \mid x < b\}$ whose leftmost character is nonzero.*

Definition: For b an integer greater than 1, w a positive integer, and n a nonnegative integer, the **base b fixed-width w expansion of n** is

$$(a_{w-1} \cdots a_1 a_0)_{b,w}$$

where $a_0, a_1, \ldots, a_{w-1}$ are nonnegative integers less than b and

$$n = \sum_{i=0}^{w-1} a_i b^i$$

Review (Week 2): signed integers

Representing negative integers in binary: Fix a positive integer width for the representation w, $w > 1$.

To represent a positive integer n	To represent a negative integer $-n$
Sign-magnitude	
$[0a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$	$[1a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$
2s complement	
$[0a_{w-2} \cdots a_0]_{2c,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$	$[1a_{w-2} \cdots a_0]_{2c,w}$, where $2^{w-1} - n = (a_{w-2} \cdots a_0)_{2,w-1}$

2s complement: For positive integer n, to represent $-n$ in 2s complement with width w,

- Calculate $2^{w-1} - n$, convert result to binary fixed-width $w - 1$, pad with leading 1, or
- Express $-n$ as a sum of powers of 2, where the leftmost 2^{w-1} is negative weight, or
- Convert n to binary fixed-width w, flip bits, add 1 (ignore overflow)

Review (Week 3): fixed-width addition and circuits

Fixed-width addition: adding one bit at time, using the usual column-by-column and carry arithmetic, and dropping the carry from the leftmost column so the result is the same width as the summands. In many cases, this gives representation of the correct value for the sum when we interpret the summands in fixed-width binary or in 2s complement.

Combinatorial circuit: In a **combinatorial circuit** (also known as a **logic circuit**), we have **logic gates** connected by **wires**. The inputs to the circuits are the values set on the input wires: possible values are 0 (low) or 1 (high). The values flow along the wires from left to right. A wire may be split into two or more wires, indicated with a filled-in circle (representing solder). Values stay the same along a wire. When one or more wires flow into a gate, the output value of that gate is computed from the input values based on the gate's definition table. Outputs of gates may become inputs to other gates.

Review (Week 3): logical operators

Logical operators: aka propositional connectives

Conjunction	AND	$\wedge$	2 inputs. Evaluates to T exactly when **both** inputs are T
Exclusive or	XOR	$\oplus$	2 inputs. Evaluates to T exactly when **exactly one** of inputs is T
Disjunction	OR	$\vee$	2 inputs. Evaluates to T exactly when **at least one** of inputs is T
Negation	NOT	$\neg$	1 input. Evaluates to T exactly when its input is F

Truth tables: Input-output tables where we use T for 1 and F for 0.

Truth tables of the logical operators:

Input		Output		
		Conjunction	**Exclusive or**	**Disjunction**
p	q	$p \wedge q$	$p \oplus q$	$p \vee q$
T	T	T	F	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F
		AND	XOR	OR

Input	Output
	Negation
p	$\neg p$
T	F
F	T
	NOT

Review (Week 3): normal forms and logical equivalence

Definition: An expression built of variables and logical operators is in **disjunctive normal form** (DNF) means that it is an OR of ANDs of variables and their negations.

Definition: An expression built of variables and logical operators is in **conjunctive normal form** (CNF) means that it is an AND of ORs of variables and their negations.

Logical equivalence: Two compound propositions are **logically equivalent** means that they have the same truth values for all settings of truth values to their propositional variables.

Tautology: A compound proposition that evaluates to true for all settings of truth values to its propositional variables; it is abbreviated T.

Contradiction: A compound proposition that evaluates to false for all settings of truth values to its propositional variables; it is abbreviated F.

Contingency: A compound proposition that is neither a tautology nor a contradiction.

(Some) logical equivalences: *Can replace p and q with any compound proposition*

$\neg(\neg p) \equiv p$		**Double negation**
$p \vee q \equiv q \vee p$	$p \wedge q \equiv q \wedge p$	**Commutativity** Ordering of terms
$(p \vee q) \vee r \equiv p \vee (q \vee r)$	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	**Associativity** Grouping of terms
$p \wedge F \equiv F \quad p \vee T \equiv T$	$p \wedge T \equiv p \quad p \vee F \equiv p$	**Domination** aka short circuit evaluation
$\neg(p \wedge q) \equiv \neg p \vee \neg q$	$\neg(p \vee q) \equiv \neg p \wedge \neg q$	**DeMorgan's Laws**

Part 1: Representing numbers

Q1. Interpreting bit strings: range and zero

(a) What are the smallest and largest integers that can be represented in sign-magnitude width 4? In 2s complement width 4?

(b) Does 0 have a base 2 expansion? Does it have a base 2 fixed-width 4 expansion? Justify using the definitions.

▸ Definitions

Q1. Interpreting bit strings: range and zero

(a) What are the smallest and largest integers that can be represented in sign-magnitude width 4? In 2s complement width 4?

Sign-magnitude: the magnitude $n = (a_2 a_1 a_0)_{2,3}$ is between 0 and 7, so the integers are $\boxed{-7 \text{ to } 7}$.

2s complement: with leftmost bit 0, $n = (a_2 a_1 a_0)_{2,3} \leq 7$. With leftmost bit 1, $8 - n = (a_2 a_1 a_0)_{2,3}$ ranges over $0, \ldots, 7$, so n ranges over $1, \ldots, 8$. The integers are $\boxed{-8 \text{ to } 7}$.

The range is not symmetric: -8 can be represented but 8 cannot.

(b) Does 0 have a base 2 expansion? Does it have a base 2 fixed-width 4 expansion? Justify using the definitions.

▸ Definitions

Q1. Interpreting bit strings: range and zero

(a) What are the smallest and largest integers that can be represented in sign-magnitude width 4? In 2s complement width 4?

Sign-magnitude: the magnitude $n = (a_2 a_1 a_0)_{2,3}$ is between 0 and 7, so the integers are $\boxed{-7 \text{ to } 7}$.

2s complement: with leftmost bit 0, $n = (a_2 a_1 a_0)_{2,3} \leq 7$. With leftmost bit 1, $8 - n = (a_2 a_1 a_0)_{2,3}$ ranges over $0, \ldots, 7$, so n ranges over $1, \ldots, 8$. The integers are $\boxed{-8 \text{ to } 7}$.

The range is not symmetric: -8 can be represented but 8 cannot.

(b) Does 0 have a base 2 expansion? Does it have a base 2 fixed-width 4 expansion? Justify using the definitions.

$\boxed{\text{No base } 2 \text{ expansion}}$: the base b expansion is defined only for a *positive* integer n (and needs $a_{k-1} \neq 0$, while any sum $\sum a_i 2^i$ with a nonzero coefficient is positive).

$\boxed{\text{Yes}}$: $(0000)_{2,4}$, since the fixed-width definition allows a nonnegative n and $0 = 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$.

▸ Definitions

Q2. Least significant first: why mod gives the coefficients

```
1  procedure basebleast(n, b: positive integers with b > 1)
2  k := 0
3  q := n
4  while q ≠ 0
5      a_k := q mod b
6      k := k + 1
7      q := q div b
8  return (a_{k-1}, ..., a_0)
```

(a) Complete the trace table for $basebleast$ with inputs $n = 17$ and $b = 3$. What is the output?

	a_{k-1}	k	q	$q \neq 0$?
Initial value				
After 1 iteration				
After 2 iterations				
After 3 iterations				

▸ Definitions

Q2. Least significant first: why mod gives the coefficients

```
1  procedure basebleast(n, b: positive integers with b > 1)
2  k := 0
3  q := n
4  while q ≠ 0
5      a_k := q mod b
6      k := k + 1
7      q := q div b
8  return (a_{k-1}, ..., a_0)
```

(a) Complete the trace table for $basebleast$ with inputs $n = 17$ and $b = 3$. What is the output?

	a_{k-1}	k	q	$q \neq 0$?
Initial value	–	0	17	T
After 1 iteration	2	1	5	T
After 2 iterations	2	2	1	T
After 3 iterations	1	3	0	F

Each iteration computes q **mod** 3 and q **div** 3: $17 = 3 \cdot 5 + 2$, $5 = 3 \cdot 1 + 2$, $1 = 3 \cdot 0 + 1$.

The output is $(a_2, a_1, a_0) = (1, 2, 2)$, so $\boxed{(122)_3}$ is the base 3 expansion of 17. Check: $1 \cdot 3^2 + 2 \cdot 3 + 2 = 17$.

▸ Definitions

Q2. Least significant first: why mod gives the coefficients

(b) Explain why the first coefficient that $basebleast$ computes, q **mod** b, is the rightmost coefficient a_0 of the base b expansion of n. Why does the algorithm then continue with q **div** b?

▸ Definitions

Q2. Least significant first: why mod gives the coefficients

(b) Explain why the first coefficient that $baseb least$ computes, q **mod** b, is the rightmost coefficient a_0 of the base b expansion of n. Why does the algorithm then continue with q **div** b?

When $k > 1$, $n = a_{k-1}b^{k-1} + \cdots + a_1 b + a_0 = b\,(a_{k-1}b^{k-2} + \cdots + a_1) + a_0$, with $0 \le a_0 < b$.

This has the form $n = dq + r$ with $d = b$ and $0 \le r < d$. By the Division Algorithm the quotient and remainder are *unique*, so

$$a_0 = n \textbf{ mod } b \qquad \text{and} \qquad a_{k-1}b^{k-2} + \cdots + a_1 = n \textbf{ div } b.$$

The quotient n **div** b has coefficients $a_{k-1}, \ldots, a_1$, so applying the same step to it gives a_1, then a_2, and so on, until the quotient is 0.

▸ Definitions

Part 2: Circuits

Q3. Negation circuits and corner cases

Inputs x, y, z, w represent $[x\,y\,z\,w]$ and outputs x', y', z', w' represent $[x'\,y'\,z'\,w']$ (width 4).

(a) Design a circuit that, given $[x\,y\,z\,w]_{s,4}$ representing an integer, outputs $[x'\,y'\,z'\,w']_{s,4}$ representing -1 times that integer. Test it with the integers 5 and 0. Are there any corner cases where it doesn't work?

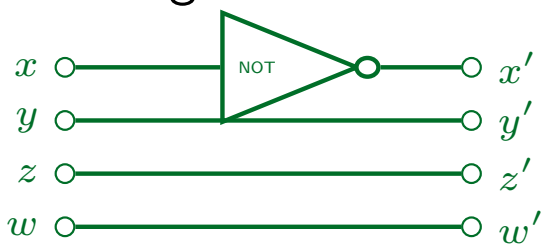
(b) Does the same circuit work for 2s complement width 4? Test it with the integer 5.

▸ Definitions

Q3. Negation circuits and corner cases

Inputs x, y, z, w represent $[x\,y\,z\,w]$ and outputs x', y', z', w' represent $[x'\,y'\,z'\,w']$ (width 4).

(a) Design a circuit that, given $[x\,y\,z\,w]_{s,4}$ representing an integer, outputs $[x'\,y'\,z'\,w']_{s,4}$ representing -1 times that integer. Test it with the integers 5 and 0. Are there any corner cases where it doesn't work?



Only the leftmost (sign) bit changes; the magnitude bits are copied.

$5 = [0101]_{s,4} \mapsto [1101]_{s,4} = -5$

$0 = [0000]_{s,4} \mapsto [1000]_{s,4}$, which also represents $0 = -0$.

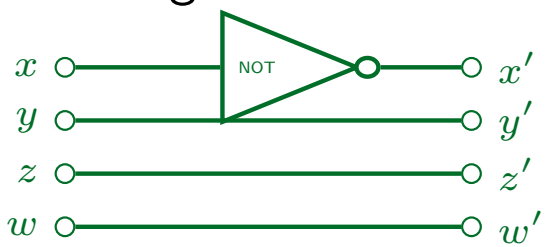
No corner cases: every input is mapped to a representation of its negative.

(b) Does the same circuit work for 2s complement width 4? Test it with the integer 5.

Q3. Negation circuits and corner cases

Inputs x, y, z, w represent $[x\,y\,z\,w]$ and outputs x', y', z', w' represent $[x'\,y'\,z'\,w']$ (width 4).

(a) Design a circuit that, given $[x\,y\,z\,w]_{s,4}$ representing an integer, outputs $[x'\,y'\,z'\,w']_{s,4}$ representing -1 times that integer. Test it with the integers 5 and 0. Are there any corner cases where it doesn't work?



Only the leftmost (sign) bit changes; the magnitude bits are copied.

$5 = [0101]_{s,4} \mapsto [1101]_{s,4} = -5$

$0 = [0000]_{s,4} \mapsto [1000]_{s,4}$, which also represents $0 = -0$.

No corner cases: every input is mapped to a representation of its negative.

(b) Does the same circuit work for 2s complement width 4? Test it with the integer 5.

No. $5 = [0101]_{2c,4} \mapsto [1101]_{2c,4}$, which represents $-n$ with $8 - n = (101)_{2,3} = 5$, i.e. -3.
But $-5 = [1011]_{2c,4}$, since $8 - 5 = 3 = (011)_{2,3}$.

Q4. Half adders and fixed-width addition

(a) Inputs x_1x_0 and y_1y_0 represent $(x_1x_0)_{2,2}$ and $(y_1y_0)_{2,2}$. In class, the outputs of the width-2 adder are

$$z_0 = x_0 \oplus y_0 \qquad z_1 = (x_1 \oplus y_1) \oplus (x_0 \wedge y_0) \qquad z_2 = (x_1 \wedge y_1) \oplus \big((x_1 \oplus y_1) \wedge (x_0 \wedge y_0)\big)$$

What does the part $x_0 \wedge y_0$ represent? Compute z_1 and z_2 for $(11)_{2,2} + (01)_{2,2}$.

(b) Fixed-width addition drops the carry from the leftmost column. For each sum below, decide whether the result represents the actual sum when the bit strings are interpreted in binary fixed-width 4, and when they are interpreted in 2s complement width 4.

$$0111 + 0001 = 1000 \qquad\qquad 1111 + 0001 = 0000$$

▸ Definitions

Q4. Half adders and fixed-width addition

(a) Inputs x_1x_0 and y_1y_0 represent $(x_1x_0)_{2,2}$ and $(y_1y_0)_{2,2}$. In class, the outputs of the width-2 adder are

$$z_0 = x_0 \oplus y_0 \qquad z_1 = (x_1 \oplus y_1) \oplus (x_0 \wedge y_0) \qquad z_2 = (x_1 \wedge y_1) \oplus \big((x_1 \oplus y_1) \wedge (x_0 \wedge y_0)\big)$$

What does the part $x_0 \wedge y_0$ represent? Compute z_1 and z_2 for $(11)_{2,2} + (01)_{2,2}$.

$x_0 \wedge y_0 = c_0$ is the carry from the right column, which is added into the left column.
$x_0 = y_0 = 1$, so $c_0 = 1$ (and $z_0 = 0$). $z_1 = (1 \oplus 0) \oplus 1 = 0$, $z_2 = (1 \wedge 0) \oplus \big((1 \oplus 0) \wedge 1\big) = 1$.
Result $(100)_{2,3} = 4 = 3 + 1$.

(b) Fixed-width addition drops the carry from the leftmost column. For each sum below, decide whether the result represents the actual sum when the bit strings are interpreted in binary fixed-width 4, and when they are interpreted in 2s complement width 4.

$$0111 + 0001 = 1000 \qquad\qquad 1111 + 0001 = 0000$$

▸ Definitions

Q4. Half adders and fixed-width addition

(a) Inputs x_1x_0 and y_1y_0 represent $(x_1x_0)_{2,2}$ and $(y_1y_0)_{2,2}$. In class, the outputs of the width-2 adder are

$$z_0 = x_0 \oplus y_0 \qquad z_1 = (x_1 \oplus y_1) \oplus (x_0 \wedge y_0) \qquad z_2 = (x_1 \wedge y_1) \oplus \big((x_1 \oplus y_1) \wedge (x_0 \wedge y_0)\big)$$

What does the part $x_0 \wedge y_0$ represent? Compute z_1 and z_2 for $(11)_{2,2} + (01)_{2,2}$.

$x_0 \wedge y_0 = c_0$ is the carry from the right column, which is added into the left column.
$x_0 = y_0 = 1$, so $c_0 = 1$ (and $z_0 = 0$). $z_1 = (1 \oplus 0) \oplus 1 = 0$, $z_2 = (1 \wedge 0) \oplus \big((1 \oplus 0) \wedge 1\big) = 1$.
Result $(100)_{2,3} = 4 = 3 + 1$.

(b) Fixed-width addition drops the carry from the leftmost column. For each sum below, decide whether the result represents the actual sum when the bit strings are interpreted in binary fixed-width 4, and when they are interpreted in 2s complement width 4.

$$0111 + 0001 = 1000 \qquad\qquad 1111 + 0001 = 0000$$

	binary fixed-width 4		2s complement width 4	
$0111 + 0001 = 1000$	$7 + 1 = 8 = (1000)_{2,4}$	correct	$7 + 1$, but $[1000]_{2c,4} = -8$	incorrect
$1111 + 0001 = 0000$	$15 + 1 = 16$, but $(0000)_{2,4} = 0$	incorrect	$[1111]_{2c,4} = -1$, $-1 + 1 = 0$	correct

The errors happen when the actual sum (8, resp. 16) is not representable. The same circuit is used in both cases; whether the result is correct depends on the interpretation.

▸ Definitions

Part 3: Propositional logic

Q5. Normal forms and logical equivalence

Input			Output
p	q	r	?
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	T

(a) Write an expression in CNF whose output column matches the table.

(b) A DNF expression for this table would use one AND for each row with output T. How many ANDs would it have? Write the AND for the row TFF.

▸ Definitions

Q5. Normal forms and logical equivalence

Input			Output
p	q	r	?
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	T

(a) Write an expression in CNF whose output column matches the table.

One OR for each row with output F, avoiding that row:
avoid TTF: $\neg p \vee \neg q \vee r$; avoid FTF: $p \vee \neg q \vee r$.

$$(\neg p \vee \neg q \vee r) \wedge (p \vee \neg q \vee r)$$

(b) A DNF expression for this table would use one AND for each row with output T. How many ANDs would it have? Write the AND for the row TFF.

▸ Definitions

Q5. Normal forms and logical equivalence

Input			Output
p	q	r	?
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	T

(a) Write an expression in CNF whose output column matches the table.

One OR for each row with output F, avoiding that row: avoid TTF: $\neg p \vee \neg q \vee r$; avoid FTF: $p \vee \neg q \vee r$.

$$\boxed{(\neg p \vee \neg q \vee r) \wedge (p \vee \neg q \vee r)}$$

(b) A DNF expression for this table would use one AND for each row with output T. How many ANDs would it have? Write the AND for the row TFF.

$\boxed{6 \text{ ANDs}}$ (six rows have output T). For the row TFF: $\boxed{p \wedge \neg q \wedge \neg r}$, which is T in that row only. Here the CNF is shorter because there are fewer F rows than T rows.

▸ Definitions

Q5. Normal forms and logical equivalence

(c) A student claims that $\neg(p \oplus q) \equiv \neg p \oplus \neg q$, "just like DeMorgan's laws". Decide whether the claim is correct and justify your answer.

▸ Definitions

Q5. Normal forms and logical equivalence

(c) A student claims that $\neg(p \oplus q) \equiv \neg p \oplus \neg q$, "just like DeMorgan's laws". Decide whether the claim is correct and justify your answer.

Incorrect. It is enough to find one row where the outputs disagree. With $p = T$, $q = T$: $\neg(T \oplus T) = \neg F = T$, but $\neg T \oplus \neg T = F \oplus F = F$. DeMorgan's laws are stated for $\wedge$ and $\vee$ only.

▸ Definitions

Appendix: Definitions from the lecture notes

Definitions for Q1

Representing negative integers in binary: Fix a positive integer width for the representation w, $w > 1$.

To represent a positive integer n	To represent a negative integer $-n$
Sign-magnitude	
$[0a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$	$[1a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$
2s complement	
$[0a_{w-2} \cdots a_0]_{2c,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$	$[1a_{w-2} \cdots a_0]_{2c,w}$, where $2^{w-1} - n = (a_{w-2} \cdots a_0)_{2,w-1}$

Back to Q1

Definitions for Q1

Definition: For b an integer greater than 1 and n a positive integer, the **base b expansion of n** is

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are (symbols for) nonnegative integers less than b, $a_{k-1} \neq 0$, and

$$n = \sum_{i=0}^{k-1} a_i b^i$$

Notice: *The base b expansion of a positive integer n is a string over the alphabet $\{x \in \mathbb{N} \mid x < b\}$ whose leftmost character is nonzero.*

◂ Back to Q1

Definitions for Q1

Definition: For b an integer greater than 1, w a positive integer, and n a nonnegative integer, the **base b fixed-width w expansion of n** is

$$(a_{w-1} \cdots a_1 a_0)_{b,w}$$

where $a_0, a_1, \ldots, a_{w-1}$ are nonnegative integers less than b and

$$n = \sum_{i=0}^{w-1} a_i b^i$$

◀ Back to Q1

Definitions for Q2

Least significant first: Start with right-most coefficient of expansion (lowest value). Idea: (when $k > 1$)

$$\begin{aligned} n &= a_{k-1}b^{k-1} + \cdots + a_1 b + a_0 \\ &= b(a_{k-1}b^{k-2} + \cdots + a_1) + a_0 \end{aligned}$$

so $a_0 = n$ **mod** b and $a_{k-1}b^{k-2} + \cdots + a_1 = n$ **div** b.
Calculating base b expansion, from right:

1 **procedure** *basebleast*(n, b: positive integers with $b > 1$)
2 $k := 0$
3 $q := n$
4 **while** $q \neq 0$
5 $\quad a_k := q$ **mod** b
6 $\quad k := k + 1$
7 $\quad q := q$ **div** b
8 **return** $(a_{k-1}, \ldots, a_0)\{(a_{k-1} \ldots a_0)_b$ is the base b expansion of $n\}$

◀ Back to Q2

Definitions for Q2

Integer division and remainders: (aka The Division Algorithm) Let n be an integer and d a positive integer. There are unique integers q and r, with $0 \leq r < d$, such that $n = dq + r$. In this case, d is called the divisor, n is called the dividend, q is called the quotient, and r is called the remainder.

div and mod: Because these numbers are guaranteed to exist, the following functions are well-defined: **div** $: \mathbb{Z} \times \mathbb{Z}^+ \to \mathbb{Z}$ given by **div** $(\ (n, d)\ )$ is the quotient when n is the dividend and d is the divisor. **mod** $: \mathbb{Z} \times \mathbb{Z}^+ \to \mathbb{Z}$ given by **mod** $(\ (n, d)\ )$ is the remainder when n is the dividend and d is the divisor. Because these functions are so important, we sometimes use the notation n **div** $d =$ **div** $(\ (n, d)\ )$ and n **mod** $d =$ **mod** $(\ (n, d)\ )$.

◀ Back to Q2

Definitions for Q3

Representing negative integers in binary: Fix a positive integer width for the representation w, $w > 1$.

To represent a positive integer n	To represent a negative integer $-n$
Sign-magnitude	
$[0a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$	$[1a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$
2s complement	
$[0a_{w-2} \cdots a_0]_{2c,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$	$[1a_{w-2} \cdots a_0]_{2c,w}$, where $2^{w-1} - n = (a_{w-2} \cdots a_0)_{2,w-1}$

Back to Q3

Definitions for Q3

2s complement: For positive integer n, to represent $-n$ in 2s complement with width w,

- Calculate $2^{w-1} - n$, convert result to binary fixed-width $w-1$, pad with leading 1, or
- Express $-n$ as a sum of powers of 2, where the leftmost 2^{w-1} is negative weight, or
- Convert n to binary fixed-width w, flip bits, add 1 (ignore overflow)

◀ Back to Q3

Definitions for Q3

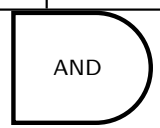
Combinatorial circuit: In a **combinatorial circuit** (also known as a **logic circuit**), we have **logic gates** connected by **wires**. The inputs to the circuits are the values set on the input wires: possible values are 0 (low) or 1 (high). The values flow along the wires from left to right. A wire may be split into two or more wires, indicated with a filled-in circle (representing solder). Values stay the same along a wire. When one or more wires flow into a gate, the output value of that gate is computed from the input values based on the gate's definition table. Outputs of gates may become inputs to other gates.

◂ Back to Q3

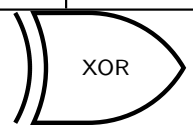
Definitions for Q3

Gate definition tables:

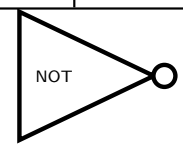
Inputs		Output
x	y	x AND y
1	1	1
1	0	0
0	1	0
0	0	0



Inputs		Output
x	y	x XOR y
1	1	0
1	0	1
0	1	1
0	0	0



Input	Output
x	NOT x
1	0
0	1



◀ Back to Q3

Definitions for Q4

Fixed-width addition: adding one bit at time, using the usual column-by-column and carry arithmetic, and dropping the carry from the leftmost column so the result is the same width as the summands. In many cases, this gives representation of the correct value for the sum when we interpret the summands in fixed-width binary or in 2s complement.

◀ Back to Q4

Definitions for Q4

Logical operators: aka propositional connectives

Conjunction	AND	$\wedge$	2 inputs. Evaluates to T exactly when **both** inputs are T
Exclusive or	XOR	$\oplus$	2 inputs. Evaluates to T exactly when **exactly one** of inputs is T
Disjunction	OR	$\vee$	2 inputs. Evaluates to T exactly when **at least one** of inputs is T
Negation	NOT	$\neg$	1 input. Evaluates to T exactly when its input is F

Truth tables: Input-output tables where we use T for 1 and F for 0.

◀ Back to Q4

Definitions for Q4

Definition: For b an integer greater than 1, w a positive integer, and n a nonnegative integer, the **base b fixed-width w expansion of n** is

$$(a_{w-1} \cdots a_1 a_0)_{b,w}$$

where $a_0, a_1, \ldots, a_{w-1}$ are nonnegative integers less than b and

$$n = \sum_{i=0}^{w-1} a_i b^i$$

◀ Back to Q4

Definitions for Q4

Representing negative integers in binary: Fix a positive integer width for the representation w, $w > 1$.

To represent a positive integer n	To represent a negative integer $-n$
Sign-magnitude	
$[0a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$	$[1a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$
2s complement	
$[0a_{w-2} \cdots a_0]_{2c,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$	$[1a_{w-2} \cdots a_0]_{2c,w}$, where $2^{w-1} - n = (a_{w-2} \cdots a_0)_{2,w-1}$

◀ Back to Q4

Definitions for Q5

Definition: An expression built of variables and logical operators is in **disjunctive normal form** (DNF) means that it is an OR of ANDs of variables and their negations.

Definition: An expression built of variables and logical operators is in **conjunctive normal form** (CNF) means that it is an AND of ORs of variables and their negations.

◀ Back to Q5

Definitions for Q5

Logical operators: aka propositional connectives

Conjunction	AND	$\wedge$	2 inputs. Evaluates to T exactly when **both** inputs are T
Exclusive or	XOR	$\oplus$	2 inputs. Evaluates to T exactly when **exactly one** of inputs is T
Disjunction	OR	$\vee$	2 inputs. Evaluates to T exactly when **at least one** of inputs is T
Negation	NOT	$\neg$	1 input. Evaluates to T exactly when its input is F

Truth tables: Input-output tables where we use T for 1 and F for 0.

◀ Back to Q5

Definitions for Q5

Logical equivalence: Two compound propositions are **logically equivalent** means that they have the same truth values for all settings of truth values to their propositional variables.

Tautology: A compound proposition that evaluates to true for all settings of truth values to its propositional variables; it is abbreviated T.

Contradiction: A compound proposition that evaluates to false for all settings of truth values to its propositional variables; it is abbreviated F.

Contingency: A compound proposition that is neither a tautology nor a contradiction.

◀ Back to Q5

Definitions for Q5

(Some) logical equivalences: *Can replace p and q with any compound proposition*

$\neg(\neg p) \equiv p$		**Double negation**
$p \vee q \equiv q \vee p$	$p \wedge q \equiv q \wedge p$	**Commutativity** Ordering of terms
$(p \vee q) \vee r \equiv p \vee (q \vee r)$	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	**Associativity** Grouping of terms
$p \wedge F \equiv F \quad p \vee T \equiv T$	$p \wedge T \equiv p \quad p \vee F \equiv p$	**Domination** aka short circuit evaluation
$\neg(p \wedge q) \equiv \neg p \vee \neg q$	$\neg(p \vee q) \equiv \neg p \wedge \neg q$	**DeMorgan's Laws**

◂ Back to Q5