# COS 126 Exam 2 Review Part 2

# Written Exam Logistics

**The second exam is on Thursday Dec. 13. Go to Exams Info page for details.**

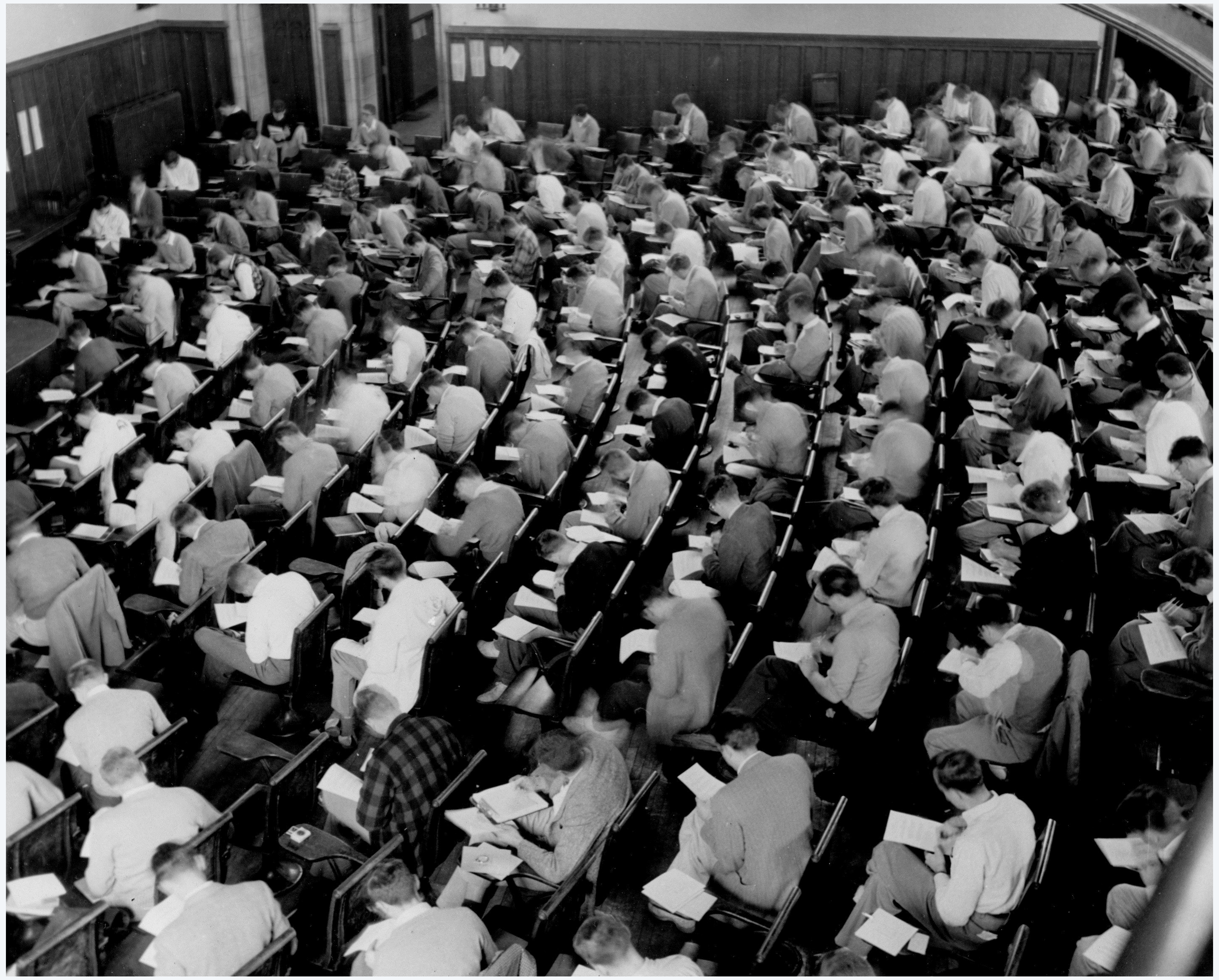
- Covers lectures since first written exam (*not* before).
- Prep session (ADTs, performance, algorithms and data structures) last week.
- Prep session (theory and combinational circuits) next.

**You don't all fit in this room.**

- Pay attention and know where to go.
- Arrive early.
- No calculator/phone/computer/headphones

**Advice.**

- Review lectures/reading.
- Try an old exam (untimed).
- Try another one (timed).
- Review a few more.



# Example question: Regular expressions

Q. *Do you understand languages and regular-expression matching ?*

**Ex.** (1990s) Consider the set of binary strings with no repeating consecutive digits: the set consisting of the empty string, **0**, **1**, **01**, **10**, **010**, **101**, **0101**, **1010**, **01010**, . . . .

Give an RE that matches strings in this set and only strings in this set.

*typical student answer*

`1 | 0 | (01)+ | (10)+`

*typical grader response*

**???**

We do not ask questions like this any more. ← **TOO DIFFICULT TO GRADE!**

## Example question: Regular expressions

Q. *Do you understand languages and regular-expression matching ?*

**Ex.** ( Fall 2011 Q4) Consider the set of binary strings with no repeating consecutive digits: the set consisting of the empty string, **0**, **1**, **01**, **10**, **010**, **101**, **0101**, **1010**, **01010**, . . . .

Fill in one circle in each row in the table at right to indicate whether the regular expression describes all strings in this set and only strings in this set (**YES**) or not (**NO**).

| | YES | NO | |
|---|---|---|---|
| **( 1 0 )* \| ( 0 1 )*** | ○ | ● | misses 0 |
| **1 \| 0 \| ( 0 1 )+ \| ( 1 0 )+** | ○ | ● | misses 0 1 0 |
| **( 0 1 )* \| ( 1 0 )* \| 0 ( 1 0 )* \| 1 ( 0 1 )*** | ● | ○ | |
| **( 1 \| 0 )* ( 0 \| 1 )*** | ○ | ● | includes 0 0 , 1 1 … |
| **( 1 \| 0 ) \| ( 0 1 \| 1 0 ) \| ( 0 1 )* \| ( 1 0 ) +** | ○ | ● | misses 0 1 0 |

# Example question: Regular expressions

Q. *Do you understand languages and regular-expression matching ?*

**Ex.** ( Fall 2014 Q5) Let L be the language **{ ab, aaab, aaaab, aabaab, aabaaab }**.
In each row, mark the column that best describes the relation between L and the given RE.

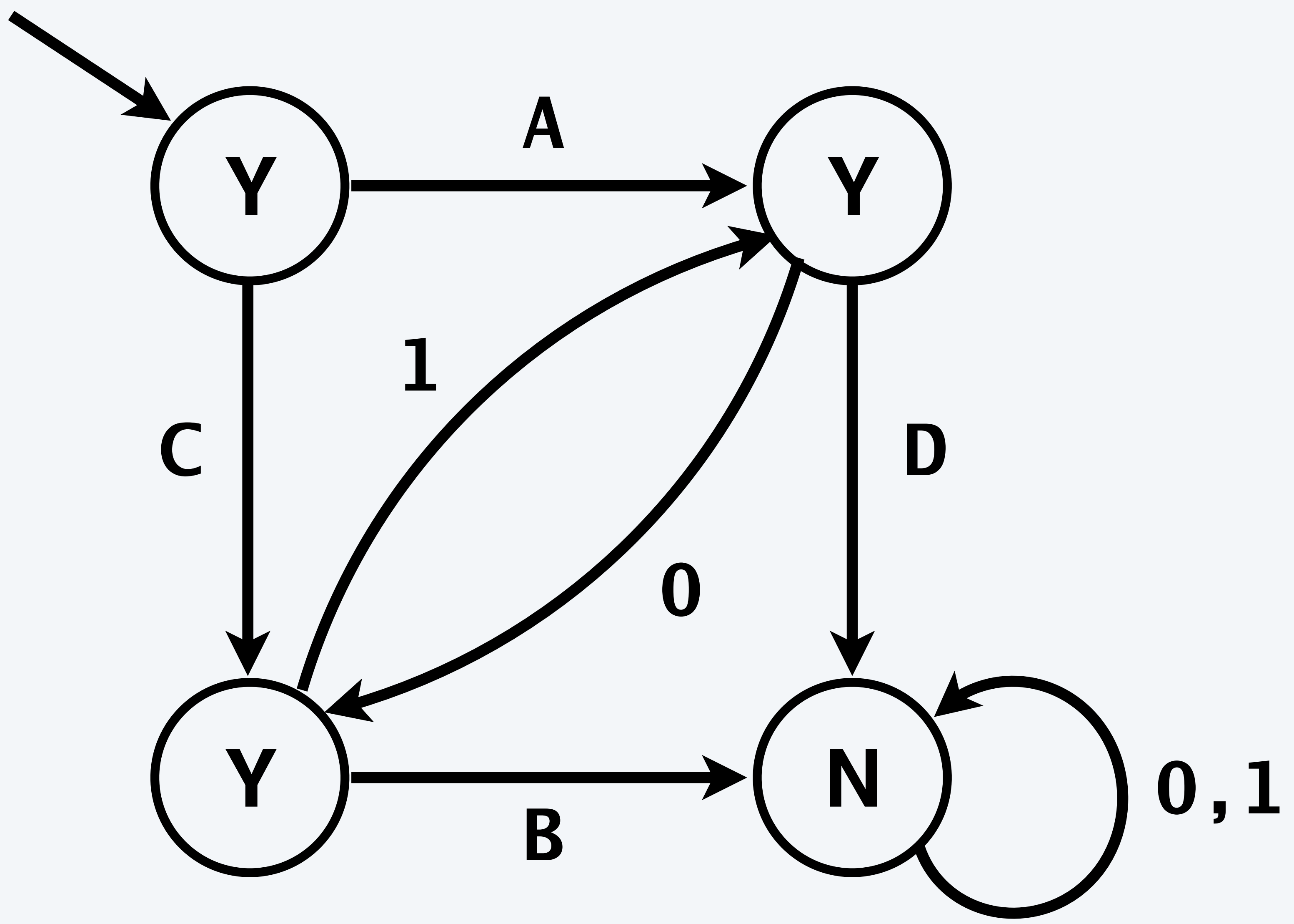
| | *matches no strings in L* | *matches some strings in L and some others* | *matches all strings in L and some others* | *matches all strings in L and no others* |
|---|---|---|---|---|
| $(aa^*b)^*$ | | | ✓ | |
| $a^*b^*$ | | ✓ | | |
| $(a \mid b)^*ab$ | | | ✓ | |
| $a^*baba^*b^*$ | ✓ | | | |
| $(ab)\mid(a(a\mid aba)(a\mid aa)b)$ | | | | ✓ |
| $a^*baaa^*b^*$ | | ✓ | | |

# Example question: DFAs

Q. *Do you understand how deterministic finite automata work ?*

**Ex.** ( Fall 2011) The DFA at left below accepts the strings in the set of binary strings with *no repeating consecutive digits* (and only those strings) but is missing four arc labels.

Fill in one circle in each row in the table at right to indicate which label (0 or 1) should replace A, B, C, and D on each of the indicated arcs.

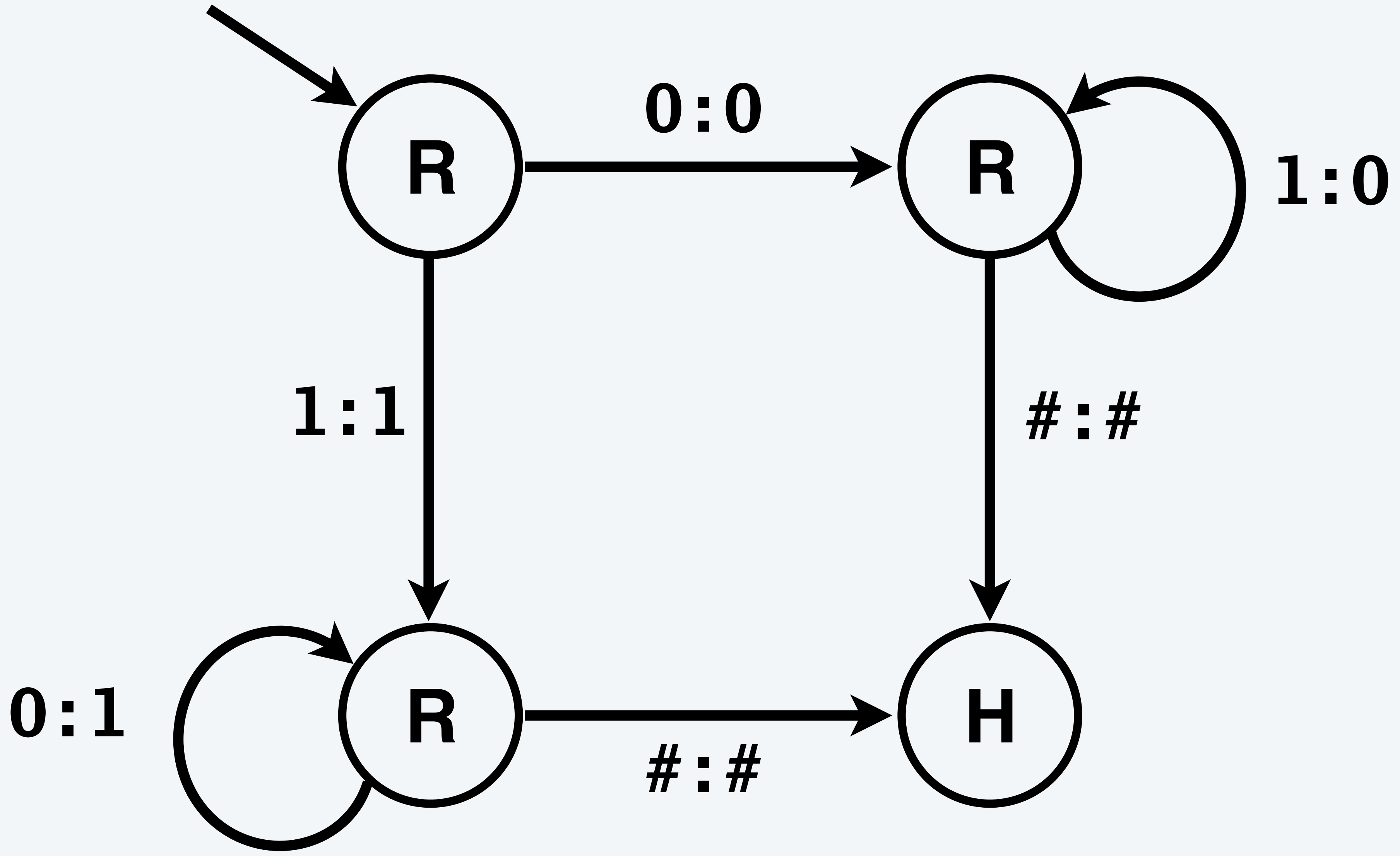



| | **0** | **1** |
|---|---|---|
| **A** | ○ | ● |
| **B** | ● | ○ |
| **C** | ● | ○ |
| **D** | ○ | ● |

# Example question: TMs

Q. *Do you understand how Turing machines work ?*

**Ex.** ( Spring 2014 [revised]) Give the result of running the TM at left for each of the given inputs. Assume that the tape head starts at the # at the left.




**# 0 1 #**

**# 0 0 #**

**# 0 1 0 0 1 1 1 0 #**

**# 0 0 0 0 0 0 0 0 #**

**# 1 1 0 0 1 1 1 0 1 #**

**# 1 1 1 1 1 1 1 1 1 #**

What does the machine do?

**Makes all bits equal to the leftmost one.**

# Example question: Intractability

Q. *Do you know basic facts about intractability ?*

**Advice:** Use your cheatsheet!

**P** is the set of all problems solvable in polynomial time.

**NP** is the set of all problems whose solutions can be *checked* in polynomial time.

**X** poly-time reduces to **Y** if using **Y** as a subroutine gives a poly-time solution to **X**.

A problem in **NP** is *NP-complete* if every problem in **NP** poly-time reduces to it.

To show that a problem in **NP** is NP-complete, show that *any* NP-complete problem poly-time reduces to it.

Factoring is in NP but not known to be NP-complete.

. . .

# Example question: Intractability

Q. *Do you know basic facts about intractability ?*

**Ex.** (Fall 2014 Q8).

| | TRUE | FALSE | ? | |
|---|---|---|---|---|
| *Every problem in NP is also in P.* | ○ | ○ | ● | **would imply P=NP** |
| *There is a DFA that can recognize all binary palindromes.* | ○ | ● | ○ | **like proof in lecture** |
| *There is a Turing machine that can decide whether the number of 1s on its input tape is prime.* | ● | ○ | ○ | **can do it in Java** |
| *The Halting Problem is NP-complete.* | ○ | ● | ○ | **not solvable, not in NP** |
| *The Traveling Salesperson Problem is NP-complete.* | ● | ○ | ○ | **poster child** |
| *There exists a deterministic Turing machine that can solve every problem in NP.* | ● | ○ | ○ | **can do it Java SLOWLY** |
| *There is a DFA that can recognize the set of all binary strings with at least one million 0s and one million1s.* | ● | ○ | ○ | **sure, a really big one** |
| *If P = NP there is a polynomial-time algorithm for factoring.* | ● | ○ | ○ | **yes, factoring is in NP** |

## Random fact: Number representation

**Ex.** (Spring 2013 Q1) Why can 3/2 be represented as an exact double in Java but 1/10 cannot?

A. *No questions on floating point representation in this exam.*

A. 3/2 = 1 + 1/2, but no way to represent 1/10 as sum of decreasing powers of 1/2.

*Confession:*
*Easier said than done !*

Note: We try to avoid asking questions on random facts nowadays.

# Random fact (or systematic application of knowledge?): Binary operations

**Ex.** (Fall 2014 Q1B) Why is **~0** equal to **-1** and not **1**?

A (wrong).

~ is "not"

0 is "false"

"not false" is “true"

"true" is 1

A (correct).

~ is BITWISE "not"

0 is 00000000000000000000000000000000

~0 is 11111111111111111111111111111111

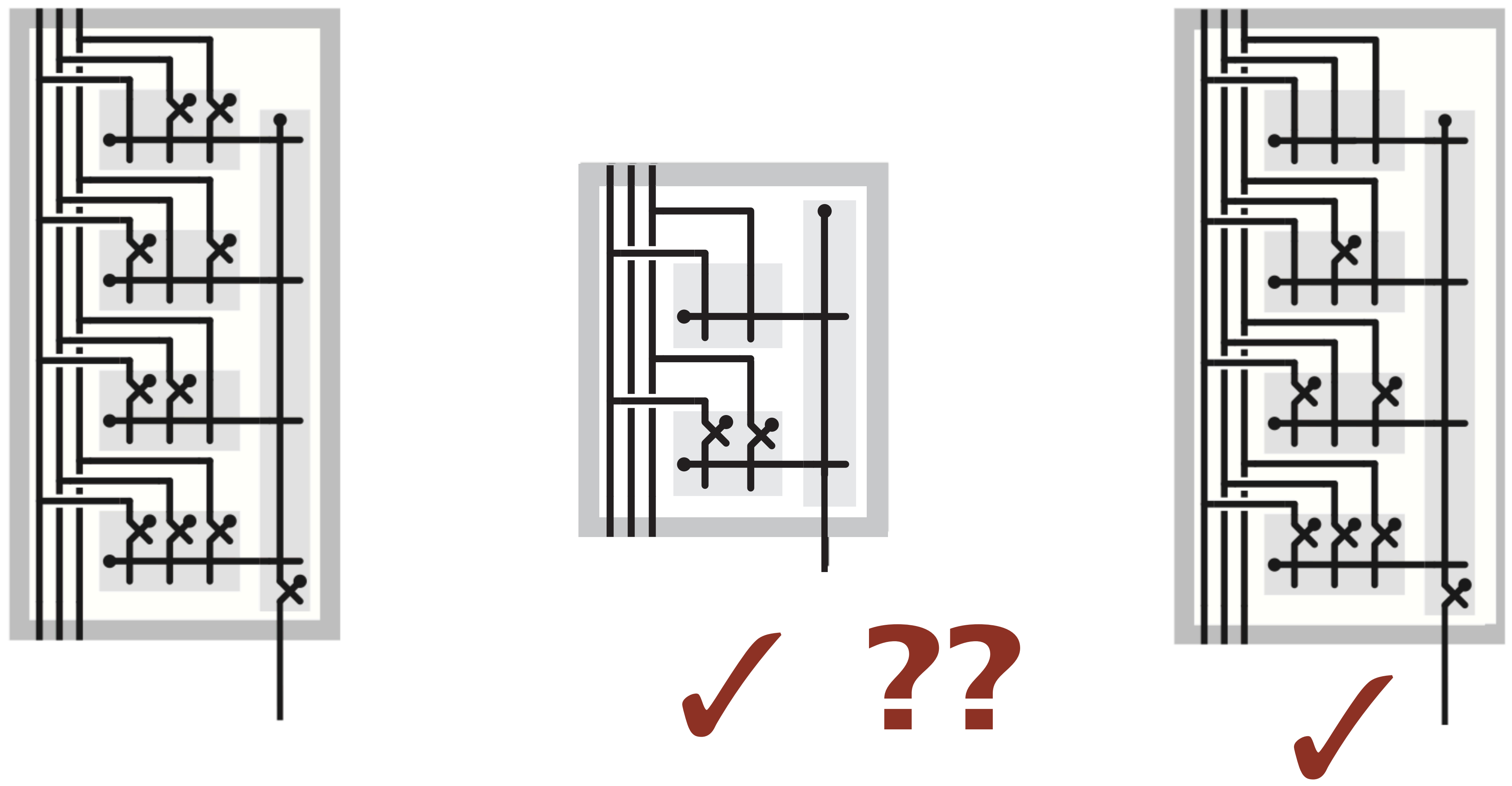
11111111111111111111111111111111 is -1 (2s complement)

# Example question: Combinational circuits

Q. *Do you understand how to build a combinational circuit that computes a Boolean function ?*

**Ex.** ( Fall 2012, Q2 [revised]) Fill in the truth table at right for the Boolean function *p* of three variables defined as follows: *p(x, y, z)* is `true` if and only if *xyz* is a palindrome (reads the same backwards or forwards). For example, 010 and 000 are palindromes, but 011 is not.

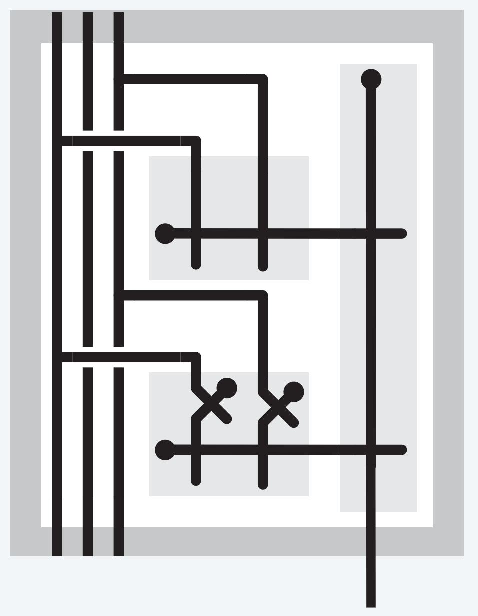
Which of the following circuits computes this function?




| *x* | *y* | *z* | *f* |
|---|---|---|---|
| 0 | 0 | 0 | **1** |
| 0 | 0 | 1 | **0** |
| 0 | 1 | 0 | **1** |
| 0 | 1 | 1 | **0** |
| 1 | 0 | 0 | **0** |
| 1 | 0 | 1 | **1** |
| 1 | 1 | 0 | **0** |
| 1 | 1 | 1 | **1** |

# Example question: Boolean logic

**Ex.** How is x'y'z' + x'yz' + xy'z + xyz reduced to x'z' + xz?



*x'z' + xz*

A. Use algebra.

```
x'y'z' + x'yz' + xy'z + xyz = x'z'(y'+ y) + xz(y'+ y)
                            = x'z'         + xz
```

A. Use a truth table.

| *x* | *y* | *z* | *f* |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

| *x* | *y* | *z* | *f* |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

| *x* | *z* | *f* |
|---|---|---|
| 0 | 0 | 1 |
| | | |
| 0 | 1 | 0 |
| | | |
| 1 | 0 | 0 |
| | | |
| 1 | 1 | 1 |
| | | |

A. Use reasoning: *xyz is a palindrome if and only if x = z (doesn't depend on y)* !

# What's next?

Written exam 2 Dec 13.

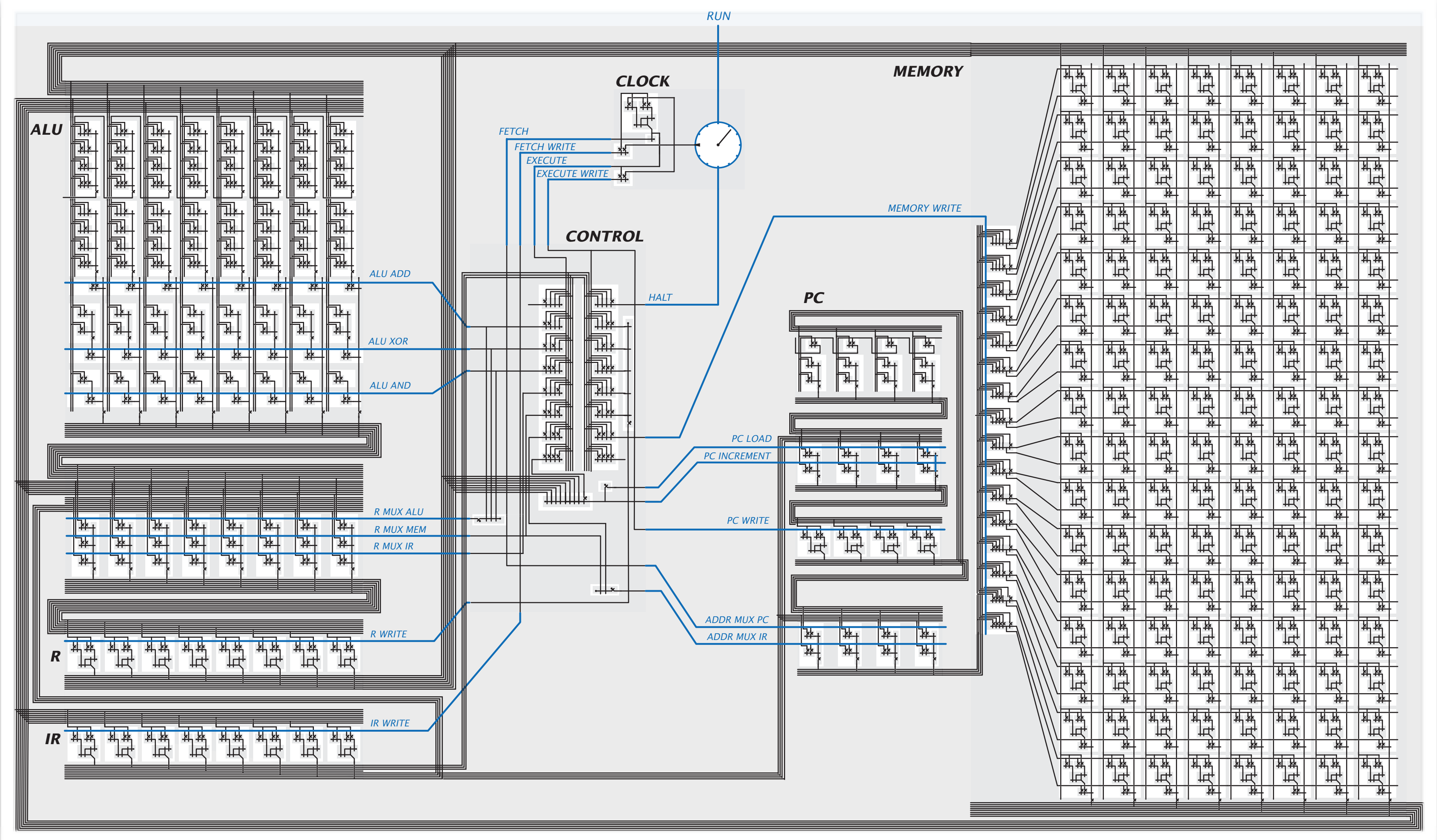
**COS 126 Written Exam 2 Fall 2018**

ATOMIC assignment due Jan 15.

1/15
**5pm**


**Atomic Nature of Matter** ( Checklist )
Re-affirm the atomic nature of matter by tracking the motion of particles undergoing Brownian motion, fitting this data to Einstein's model, and estimating Avogadro's number.

Lecture 20. How does your computer work?




Lecture 10. Programming languages.

# What's next?

Invent the future!


*“ The best way to predict the future is to invent it."*

*– Alan Kay, 1971*