

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
The in-class quizzes are in place of midterm exams.

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
The final is Wednesday, 19 December, 9-11am.

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
*The final is Wednesday, 19 December, 9-11am.
Instead of a review sheet, we have:*

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
*The final is Wednesday, 19 December, 9-11am.
Instead of a review sheet, we have:*
 - ▶ *All previous final exams (and answer keys) on the website.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
*The final is Wednesday, 19 December, 9-11am.
Instead of a review sheet, we have:*
 - ▶ *All previous final exams (and answer keys) on the website.*
 - ▶ *We'll have a mock exam during the last lecture (12 December).*

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries— we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
*The final is Wednesday, 19 December, 9-11am.
Instead of a review sheet, we have:*
 - ▶ *All previous final exams (and answer keys) on the website.*
 - ▶ *We'll have a mock exam during the last lecture (12 December).*
 - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries— we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
*The final is Wednesday, 19 December, 9-11am.
Instead of a review sheet, we have:*
 - ▶ *All previous final exams (and answer keys) on the website.*
 - ▶ *We'll have a mock exam during the last lecture (12 December).*
 - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*
 - ★ *We will use the sign-out sheets to give credit for the lecture slip.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries— we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
*The final is Wednesday, 19 December, 9-11am.
Instead of a review sheet, we have:*
 - ▶ *All previous final exams (and answer keys) on the website.*
 - ▶ *We'll have a mock exam during the last lecture (12 December).*
 - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*
 - ★ *We will use the sign-out sheets to give credit for the lecture slip.*
 - ★ *Answer key will be available after lecture.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Help! Binary & hexadecimal numbers make no sense!
No worries— we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
*The final is Wednesday, 19 December, 9-11am.
Instead of a review sheet, we have:*
 - ▶ *All previous final exams (and answer keys) on the website.*
 - ▶ *We'll have a mock exam during the last lecture (12 December).*
 - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*
 - ★ *We will use the sign-out sheets to give credit for the lecture slip.*
 - ★ *Answer key will be available after lecture.*
- Can we do more on design patterns?

Frequently Asked Questions

From lecture slips & recitation sections.

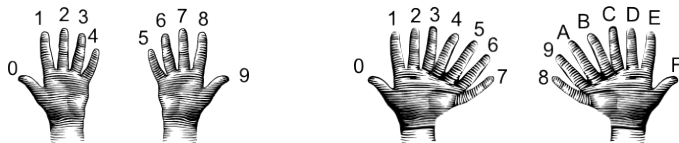
- Help! Binary & hexadecimal numbers make no sense!
No worries– we'll start off with those in today's lecture.
- Why can't in-class quizzes taken at midnight count towards my grade?
*The in-class quizzes are in place of midterm exams.
We want to know that you (and not friend/family) took the in-class quiz.*
- When is the final? Is there a review sheet?
*The final is Wednesday, 19 December, 9-11am.
Instead of a review sheet, we have:*
 - ▶ *All previous final exams (and answer keys) on the website.*
 - ▶ *We'll have a mock exam during the last lecture (12 December).*
 - ★ *Same format as real final: seating alphabetically, sign-in/sign-out, note sheet, no devices such as computers, calculators, phones, or watches.*
 - ★ *We will use the sign-out sheets to give credit for the lecture slip.*
 - ★ *Answer key will be available after lecture.*
- Can we do more on design patterns?
Yes, but we're going to transition to C++ after Thanksgiving.

Today's Topics



- Data Representation
- Machine Language: Jumps & Loops
- Recap of Python & Circuits
- Design Patterns: Sorting

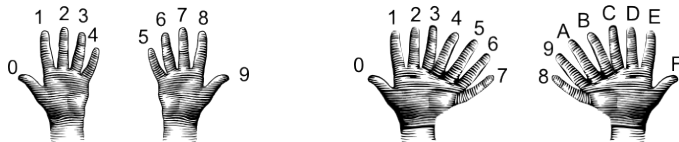
Recap: Converting between bases



(from i-programmer.info)

- From decimal to hexadecimal:

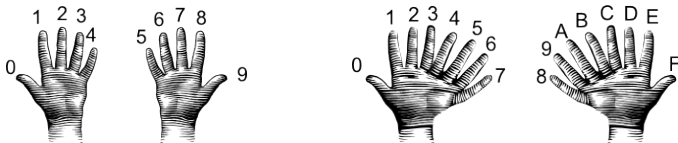
Recap: Converting between bases



(from i-programmer.info)

- From decimal to hexadecimal:
 - ▶ Divide by 16.

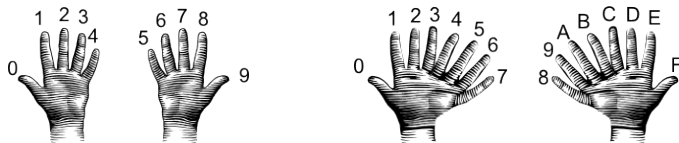
Recap: Converting between bases



(from i-programmer.info)

- From decimal to hexadecimal:
 - ▶ Divide by 16.
 - ▶ Convert quotient and remainder into hex digits.

Recap: Converting between bases

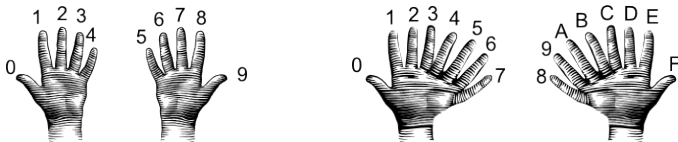


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.

Recap: Converting between bases

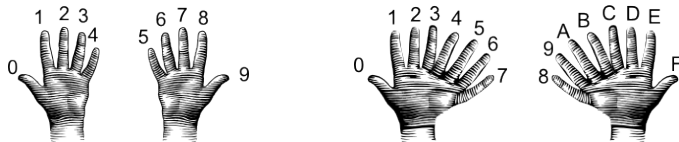


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?

Recap: Converting between bases

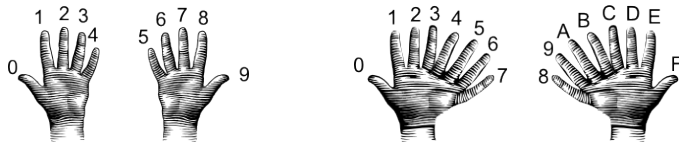


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.

Recap: Converting between bases

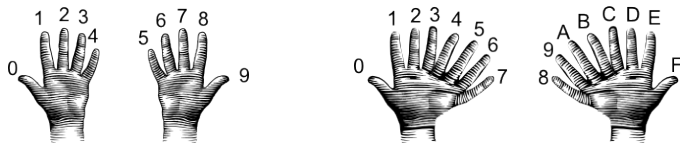


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.
12 in hex digits is C.

Recap: Converting between bases

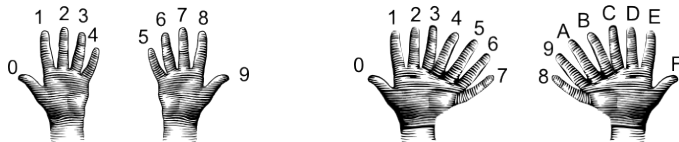


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.
12 in hex digits is C. 8 in hex digits is 8.

Recap: Converting between bases

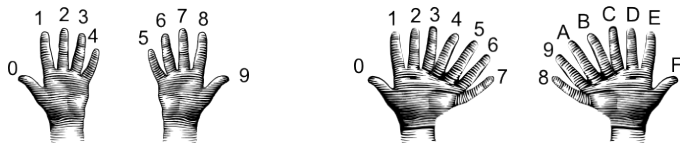


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.
12 in hex digits is C. 8 in hex digits is 8.
Answer is C8.

Recap: Converting between bases

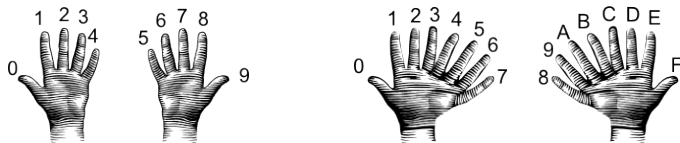


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.
12 in hex digits is C. 8 in hex digits is 8.
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?

Recap: Converting between bases

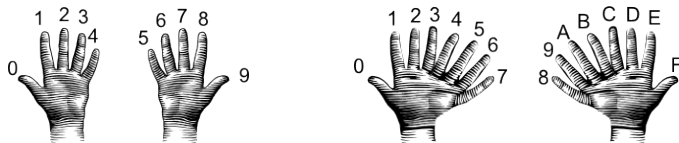


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.
12 in hex digits is C. 8 in hex digits is 8.
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?
31/16 is 1 remainder 15.

Recap: Converting between bases

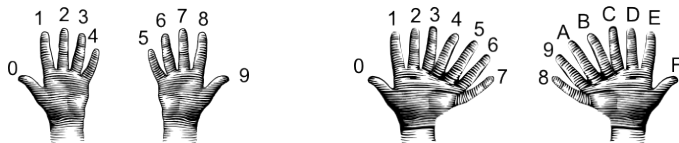


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.
12 in hex digits is C. 8 in hex digits is 8.
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?
31/16 is 1 remainder 15.
1 in hex digits is 1.

Recap: Converting between bases

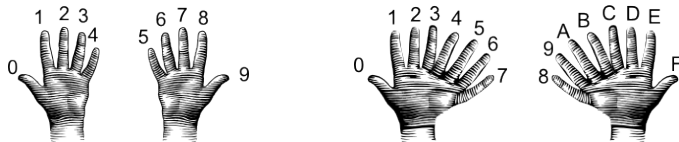


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.
12 in hex digits is C. 8 in hex digits is 8.
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?
31/16 is 1 remainder 15.
1 in hex digits is 1. 15 in hex digits is F.

Recap: Converting between bases

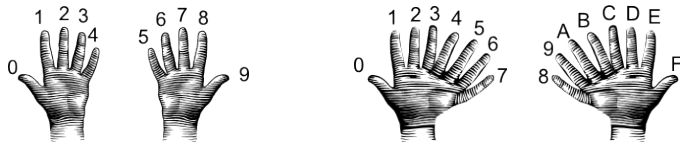


(from i-programmer.info)

- From decimal to hexadecimal:

- ▶ Divide by 16.
- ▶ Convert quotient and remainder into hex digits.
- ▶ Write down in that order to give hex notation.
- ▶ Example: what is 200 in hexadecimal notation?
200/16 is 12 remainder 8.
12 in hex digits is C. 8 in hex digits is 8.
Answer is C8.
- ▶ Example: what is 31 in hexadecimal notation?
31/16 is 1 remainder 15.
1 in hex digits is 1. 15 in hex digits is F.
Answer is 1F.

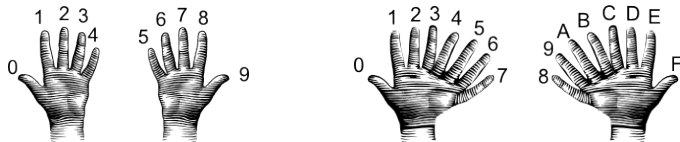
Recap: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:
 - ▶ Convert first digit to decimal and multiple by 16.

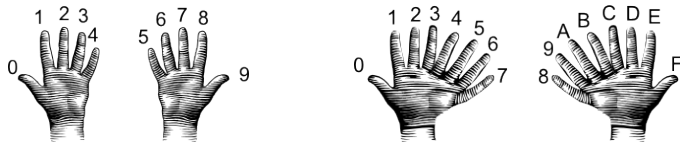
Recap: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:
 - ▶ Convert first digit to decimal and multiple by 16.
 - ▶ Convert second digit to decimal and add to total.

Recap: Converting Between Bases

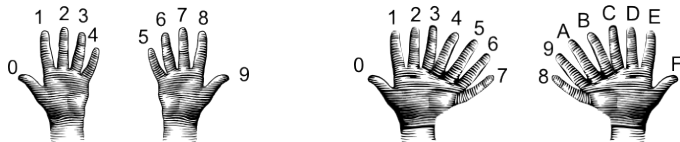


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?

Recap: Converting Between Bases

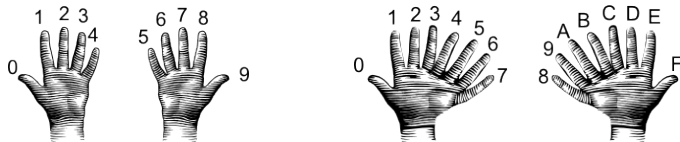


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
2 in decimal is 2.

Recap: Converting Between Bases

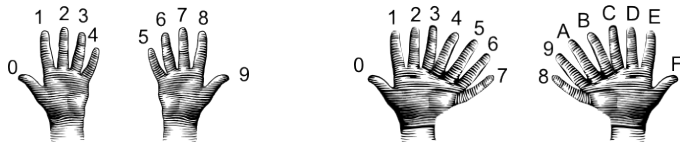


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
2 in decimal is 2. 2×16 is 32.

Recap: Converting Between Bases

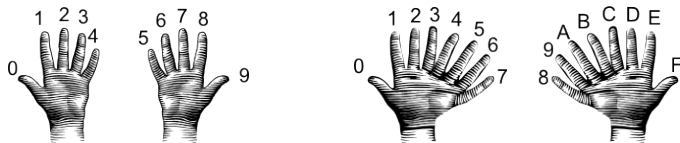


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
 - 2 in decimal is 2. 2×16 is 32.
 - A in decimal digits is 10.

Recap: Converting Between Bases

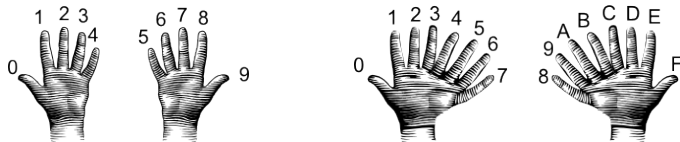


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
 - 2 in decimal is 2. 2×16 is 32.
 - A in decimal digits is 10.
 - $32 + 10$ is 42.

Recap: Converting Between Bases

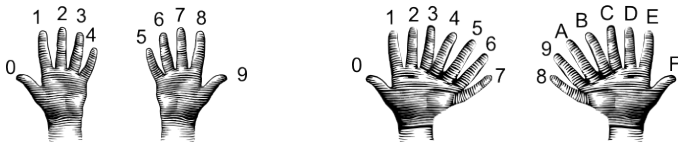


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
2 in decimal is 2. 2×16 is 32.
A in decimal digits is 10.
 $32 + 10$ is 42.
Answer is 42.
- ▶ Example: what is 99 as a decimal number?

Recap: Converting Between Bases

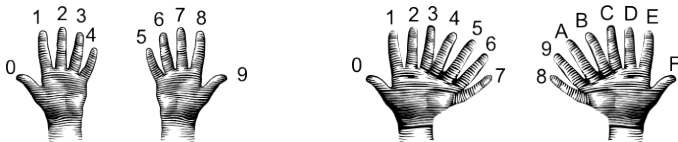


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
 - 2 in decimal is 2. 2×16 is 32.
 - A in decimal digits is 10.
 - $32 + 10$ is 42.
 - Answer is 42.
- ▶ Example: what is 99 as a decimal number?
 - 9 in decimal is 9.

Recap: Converting Between Bases

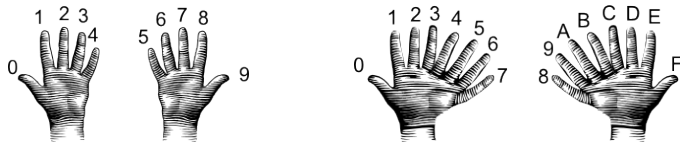


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
2 in decimal is 2. 2×16 is 32.
A in decimal digits is 10.
 $32 + 10$ is 42.
Answer is 42.
- ▶ Example: what is 99 as a decimal number?
9 in decimal is 9. 9×16 is 144.

Recap: Converting Between Bases

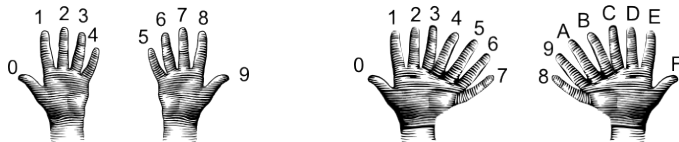


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
 - 2 in decimal is 2. 2×16 is 32.
 - A in decimal digits is 10.
 - $32 + 10$ is 42.
 - Answer is 42.
- ▶ Example: what is 99 as a decimal number?
 - 9 in decimal is 9. 9×16 is 144.
 - 9 in decimal digits is 9

Recap: Converting Between Bases

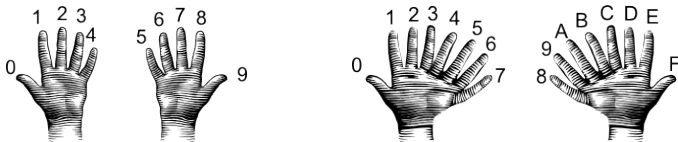


(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?
 - 2 in decimal is 2. 2×16 is 32.
 - A in decimal digits is 10.
 - $32 + 10$ is 42.
 - Answer is 42.
- ▶ Example: what is 99 as a decimal number?
 - 9 in decimal is 9. 9×16 is 144.
 - 9 in decimal digits is 9
 - $144 + 9$ is 153.

Recap: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

- ▶ Convert first digit to decimal and multiple by 16.
- ▶ Convert second digit to decimal and add to total.
- ▶ Example: what is 2A as a decimal number?

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

Answer is 42.

- ▶ Example: what is 99 as a decimal number?

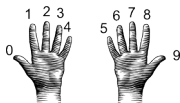
9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Answer is 153.

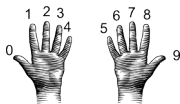
Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.

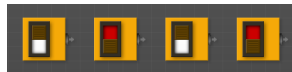
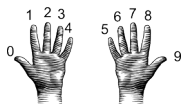
Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.

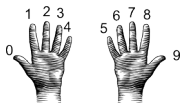
Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.

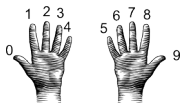
Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.

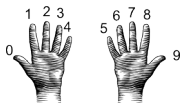
Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.

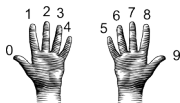
Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.

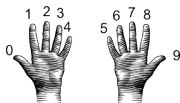
Recap: Converting Between Bases



● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.

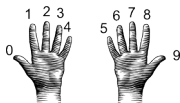
Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.

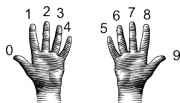
Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

Recap: Converting Between Bases

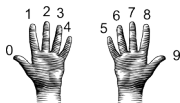


● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2.

Recap: Converting Between Bases

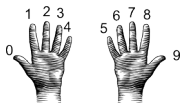


● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1:

Recap: Converting Between Bases



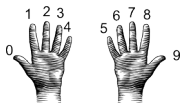
- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2.

Recap: Converting Between Bases



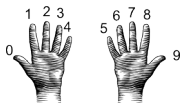
● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0:

Recap: Converting Between Bases



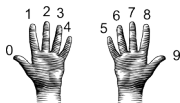
● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

Recap: Converting Between Bases



● From decimal to binary:

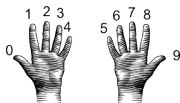
- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2.

Recap: Converting Between Bases



- From decimal to binary:

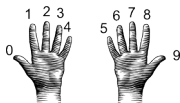
- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0:

Recap: Converting Between Bases



● From decimal to binary:

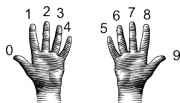
- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

Recap: Converting Between Bases



● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

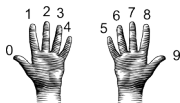
130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

2/16 is 0 rem 2.

Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

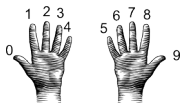
130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

2/16 is 0 rem 2. Next digit is 0:

Recap: Converting Between Bases

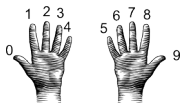


● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...

Recap: Converting Between Bases

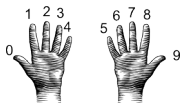


● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...
2/8 is 0 rem 2.

Recap: Converting Between Bases

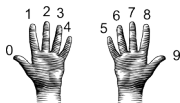


● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...
2/8 is 0 rem 2. Next digit is 0:

Recap: Converting Between Bases

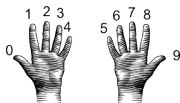


- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...
2/8 is 0 rem 2. Next digit is 0: 10000...

Recap: Converting Between Bases

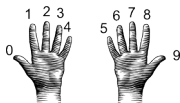


- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...
2/8 is 0 rem 2. Next digit is 0: 10000...
2/4 is 0 remainder 2.

Recap: Converting Between Bases

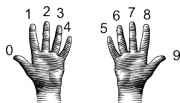


- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...
2/8 is 0 rem 2. Next digit is 0: 10000...
2/4 is 0 remainder 2. Next digit is 0:

Recap: Converting Between Bases

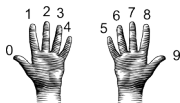


- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...
2/8 is 0 rem 2. Next digit is 0: 10000...
2/4 is 0 remainder 2. Next digit is 0: 100000...

Recap: Converting Between Bases

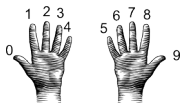


● From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...
2/8 is 0 rem 2. Next digit is 0: 10000...
2/4 is 0 remainder 2. Next digit is 0: 100000...
2/2 is 1 rem 0.

Recap: Converting Between Bases

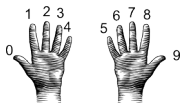


- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0: 10...
2/32 is 0 rem 2. Next digit is 0: 100...
2/16 is 0 rem 2. Next digit is 0: 1000...
2/8 is 0 rem 2. Next digit is 0: 10000...
2/4 is 0 remainder 2. Next digit is 0: 100000...
2/2 is 1 rem 0. Next digit is 1:

Recap: Converting Between Bases

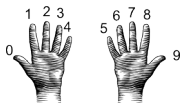


- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

```
130/128 is 1 rem 2. First digit is 1: 1...
2/64 is 0 rem 2. Next digit is 0:    10...
2/32 is 0 rem 2. Next digit is 0:    100...
2/16 is 0 rem 2. Next digit is 0:    1000...
2/8 is 0 rem 2. Next digit is 0:     10000...
2/4 is 0 remainder 2. Next digit is 0: 100000...
2/2 is 1 rem 0. Next digit is 1:     1000001...
```

Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

2/16 is 0 rem 2. Next digit is 0: 1000...

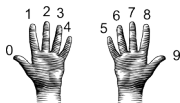
2/8 is 0 rem 2. Next digit is 0: 10000...

2/4 is 0 remainder 2. Next digit is 0: 100000...

2/2 is 1 rem 0. Next digit is 1: 1000001...

Adding the last remainder: 10000010

Recap: Converting Between Bases



- From decimal to binary:

- ▶ Divide by 128 ($= 2^7$). Quotient is the first digit.
- ▶ Divide remainder by 64 ($= 2^6$). Quotient is the next digit.
- ▶ Divide remainder by 32 ($= 2^5$). Quotient is the next digit.
- ▶ Divide remainder by 16 ($= 2^4$). Quotient is the next digit.
- ▶ Divide remainder by 8 ($= 2^3$). Quotient is the next digit.
- ▶ Divide remainder by 4 ($= 2^2$). Quotient is the next digit.
- ▶ Divide remainder by 2 ($= 2^1$). Quotient is the next digit.
- ▶ The last remainder is the last digit.
- ▶ Example: what is 130 in binary notation?

130/128 is 1 rem 2. First digit is 1: 1...

2/64 is 0 rem 2. Next digit is 0: 10...

2/32 is 0 rem 2. Next digit is 0: 100...

2/16 is 0 rem 2. Next digit is 0: 1000...

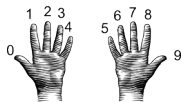
2/8 is 0 rem 2. Next digit is 0: 10000...

2/4 is 0 remainder 2. Next digit is 0: 100000...

2/2 is 1 rem 0. Next digit is 1: 1000001...

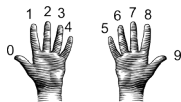
Adding the last remainder: 10000010

Recap: Converting Between Bases



- Example: what is 99 in binary notation?

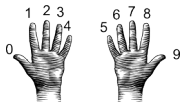
Recap: Converting Between Bases



- Example: what is 99 in binary notation?

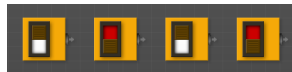
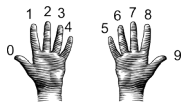
99/128 is 0 rem 99.

Recap: Converting Between Bases



- Example: what is 99 in binary notation?
99/128 is 0 rem 99. First digit is 0:

Recap: Converting Between Bases

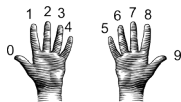


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35.

Recap: Converting Between Bases

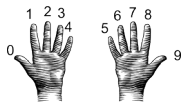


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1:

Recap: Converting Between Bases

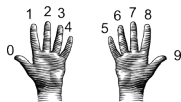


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

Recap: Converting Between Bases



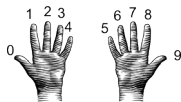
- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3.

Recap: Converting Between Bases



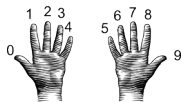
- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1:

Recap: Converting Between Bases



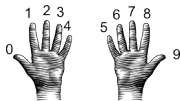
- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

Recap: Converting Between Bases



- Example: what is 99 in binary notation?

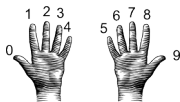
99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3.

Recap: Converting Between Bases



- Example: what is 99 in binary notation?

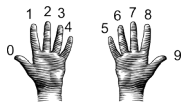
99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3. Next digit is 0:

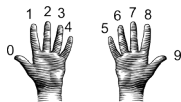
Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...
99/64 is 1 rem 35. Next digit is 1: 01...
35/32 is 1 rem 3. Next digit is 1: 011...
3/16 is 0 rem 3. Next digit is 0: 0110...

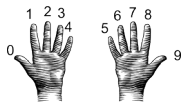
Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...
99/64 is 1 rem 35. Next digit is 1: 01...
35/32 is 1 rem 3. Next digit is 1: 011...
3/16 is 0 rem 3. Next digit is 0: 0110...
3/8 is 0 rem 3.

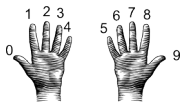
Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...
99/64 is 1 rem 35. Next digit is 1: 01...
35/32 is 1 rem 3. Next digit is 1: 011...
3/16 is 0 rem 3. Next digit is 0: 0110...
3/8 is 0 rem 3. Next digit is 0:

Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

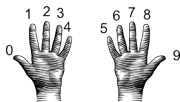
99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

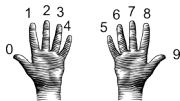
Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...
99/64 is 1 rem 35. Next digit is 1: 01...
35/32 is 1 rem 3. Next digit is 1: 011...
3/16 is 0 rem 3. Next digit is 0: 0110...
3/8 is 0 rem 3. Next digit is 0: 01100...
3/4 is 0 remainder 3.

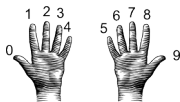
Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...
99/64 is 1 rem 35. Next digit is 1: 01...
35/32 is 1 rem 3. Next digit is 1: 011...
3/16 is 0 rem 3. Next digit is 0: 0110...
3/8 is 0 rem 3. Next digit is 0: 01100...
3/4 is 0 remainder 3. Next digit is 0:

Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

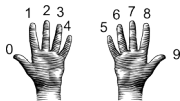
35/32 is 1 rem 3. Next digit is 1: 011...

3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

3/4 is 0 remainder 3. Next digit is 0: 011000...

Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

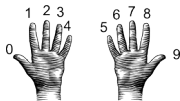
3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

3/4 is 0 remainder 3. Next digit is 0: 011000...

3/2 is 1 rem 1.

Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

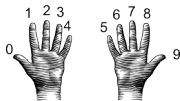
3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

3/4 is 0 remainder 3. Next digit is 0: 011000...

3/2 is 1 rem 1. Next digit is 1:

Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0: 0...

99/64 is 1 rem 35. Next digit is 1: 01...

35/32 is 1 rem 3. Next digit is 1: 011...

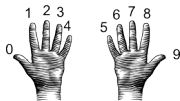
3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3. Next digit is 0: 01100...

3/4 is 0 remainder 3. Next digit is 0: 011000...

3/2 is 1 rem 1. Next digit is 1: 0110001...

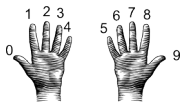
Recap: Converting Between Bases



- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0:	0...
99/64 is 1 rem 35. Next digit is 1:	01...
35/32 is 1 rem 3. Next digit is 1:	011...
3/16 is 0 rem 3. Next digit is 0:	0110...
3/8 is 0 rem 3. Next digit is 0:	01100...
3/4 is 0 remainder 3. Next digit is 0:	011000...
3/2 is 1 rem 1. Next digit is 1:	0110001...
Adding the last remainder:	01100011

Recap: Converting Between Bases

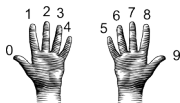


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0:	0...
99/64 is 1 rem 35. Next digit is 1:	01...
35/32 is 1 rem 3. Next digit is 1:	011...
3/16 is 0 rem 3. Next digit is 0:	0110...
3/8 is 0 rem 3. Next digit is 0:	01100...
3/4 is 0 remainder 3. Next digit is 0:	011000...
3/2 is 1 rem 1. Next digit is 1:	0110001...
Adding the last remainder:	01100011

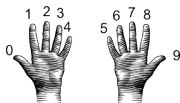
Answer is 1100011.

Recap: Converting Between Bases



- From binary to decimal:
 - ▶ Set sum = last digit.

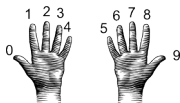
Recap: Converting Between Bases



- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.

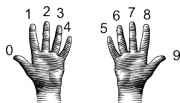
Recap: Converting Between Bases



- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.

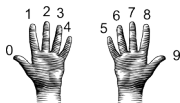
Recap: Converting Between Bases



- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.

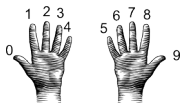
Recap: Converting Between Bases



- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.

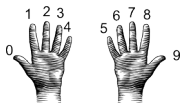
Recap: Converting Between Bases



- From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.

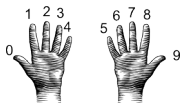
Recap: Converting Between Bases



● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.

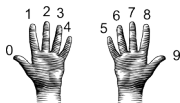
Recap: Converting Between Bases



● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.

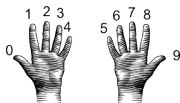
Recap: Converting Between Bases



● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.

Recap: Converting Between Bases

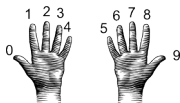


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:

Recap: Converting Between Bases



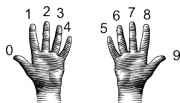
● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1

$0 \times 2 = 0$. Add 0 to sum:

Recap: Converting Between Bases



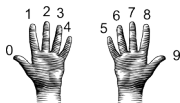
● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1

$0 \times 2 = 0$. Add 0 to sum: 1

Recap: Converting Between Bases

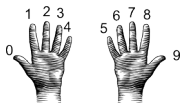


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1
 $0 \times 2 = 0$. Add 0 to sum: 1
 $1 \times 4 = 4$. Add 4 to sum:

Recap: Converting Between Bases

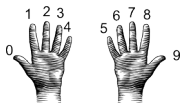


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1
 $0 \times 2 = 0$. Add 0 to sum: 1
 $1 \times 4 = 4$. Add 4 to sum: 5

Recap: Converting Between Bases

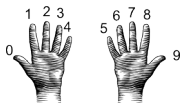


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$. Add 0 to sum:	1
$1 \times 4 = 4$. Add 4 to sum:	5
$1 \times 8 = 8$. Add 8 to sum:	

Recap: Converting Between Bases

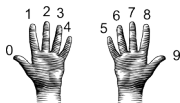


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$. Add 0 to sum:	1
$1 \times 4 = 4$. Add 4 to sum:	5
$1 \times 8 = 8$. Add 8 to sum:	13

Recap: Converting Between Bases

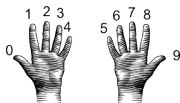


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with: 1
 $0 \times 2 = 0$. Add 0 to sum: 1
 $1 \times 4 = 4$. Add 4 to sum: 5
 $1 \times 8 = 8$. Add 8 to sum: 13
 $1 \times 16 = 16$. Add 16 to sum:

Recap: Converting Between Bases

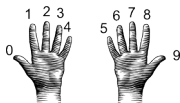


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$. Add 0 to sum:	1
$1 \times 4 = 4$. Add 4 to sum:	5
$1 \times 8 = 8$. Add 8 to sum:	13
$1 \times 16 = 16$. Add 16 to sum:	29

Recap: Converting Between Bases

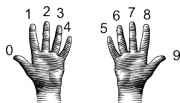


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$. Add 0 to sum:	1
$1 \times 4 = 4$. Add 4 to sum:	5
$1 \times 8 = 8$. Add 8 to sum:	13
$1 \times 16 = 16$. Add 16 to sum:	29
$1 \times 32 = 32$. Add 32 to sum:	

Recap: Converting Between Bases

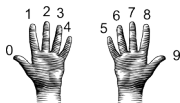


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$. Add 0 to sum:	1
$1 \times 4 = 4$. Add 4 to sum:	5
$1 \times 8 = 8$. Add 8 to sum:	13
$1 \times 16 = 16$. Add 16 to sum:	29
$1 \times 32 = 32$. Add 32 to sum:	61

Recap: Converting Between Bases

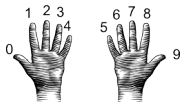


● From binary to decimal:

- ▶ Set sum = last digit.
- ▶ Multiply next digit by $2 = 2^1$. Add to sum.
- ▶ Multiply next digit by $4 = 2^2$. Add to sum.
- ▶ Multiply next digit by $8 = 2^3$. Add to sum.
- ▶ Multiply next digit by $16 = 2^4$. Add to sum.
- ▶ Multiply next digit by $32 = 2^5$. Add to sum.
- ▶ Multiply next digit by $64 = 2^6$. Add to sum.
- ▶ Multiply next digit by $128 = 2^7$. Add to sum.
- ▶ Sum is the decimal number.
- ▶ Example: What is 111101 in decimal?

Sum starts with:	1
$0 \times 2 = 0$. Add 0 to sum:	1
$1 \times 4 = 4$. Add 4 to sum:	5
$1 \times 8 = 8$. Add 8 to sum:	13
$1 \times 16 = 16$. Add 16 to sum:	29
$1 \times 32 = 32$. Add 32 to sum:	61

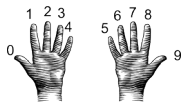
Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:

Recap: Converting Between Bases

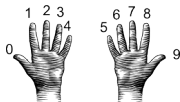


- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 * 2 = 0$. Add 0 to sum:

Recap: Converting Between Bases

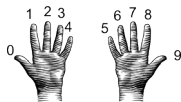


- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 * 2 = 0$. Add 0 to sum: 0

Recap: Converting Between Bases



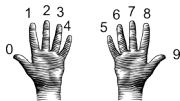
- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 \times 2 = 0$. Add 0 to sum: 0

$1 \times 4 = 4$. Add 4 to sum:

Recap: Converting Between Bases



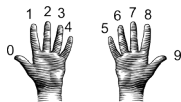
- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 \times 2 = 0$. Add 0 to sum: 0

$1 \times 4 = 4$. Add 4 to sum: 4

Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

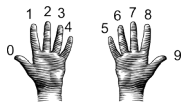
Sum starts with: 0

$0 \times 2 = 0$. Add 0 to sum: 0

$1 \times 4 = 4$. Add 4 to sum: 4

$0 \times 8 = 0$. Add 0 to sum:

Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

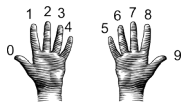
Sum starts with: 0

$0 \times 2 = 0$. Add 0 to sum: 0

$1 \times 4 = 4$. Add 4 to sum: 4

$0 \times 8 = 0$. Add 0 to sum: 4

Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

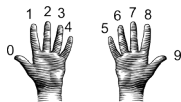
$0 \times 2 = 0$. Add 0 to sum: 0

$1 \times 4 = 4$. Add 4 to sum: 4

$0 \times 8 = 0$. Add 0 to sum: 4

$0 \times 16 = 0$. Add 0 to sum:

Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

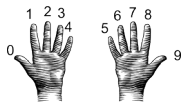
$0 \times 2 = 0$. Add 0 to sum: 0

$1 \times 4 = 4$. Add 4 to sum: 4

$0 \times 8 = 0$. Add 0 to sum: 4

$0 \times 16 = 0$. Add 0 to sum: 4

Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

$0 \times 2 = 0$. Add 0 to sum: 0

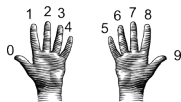
$1 \times 4 = 4$. Add 4 to sum: 4

$0 \times 8 = 0$. Add 0 to sum: 4

$0 \times 16 = 0$. Add 0 to sum: 4

$1 \times 32 = 32$. Add 32 to sum:

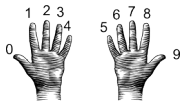
Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$. Add 0 to sum:	0
$1 \times 4 = 4$. Add 4 to sum:	4
$0 \times 8 = 0$. Add 0 to sum:	4
$0 \times 16 = 0$. Add 0 to sum:	4
$1 \times 32 = 32$. Add 32 to sum:	36

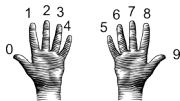
Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$. Add 0 to sum:	0
$1 \times 4 = 4$. Add 4 to sum:	4
$0 \times 8 = 0$. Add 0 to sum:	4
$0 \times 16 = 0$. Add 0 to sum:	4
$1 \times 32 = 32$. Add 32 to sum:	36
$0 \times 64 = 0$. Add 0 to sum:	

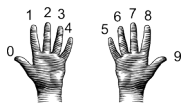
Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$. Add 0 to sum:	0
$1 \times 4 = 4$. Add 4 to sum:	4
$0 \times 8 = 0$. Add 0 to sum:	4
$0 \times 16 = 0$. Add 0 to sum:	4
$1 \times 32 = 32$. Add 32 to sum:	36
$0 \times 64 = 0$. Add 0 to sum:	36

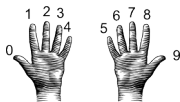
Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$. Add 0 to sum:	0
$1 \times 4 = 4$. Add 4 to sum:	4
$0 \times 8 = 0$. Add 0 to sum:	4
$0 \times 16 = 0$. Add 0 to sum:	4
$1 \times 32 = 32$. Add 32 to sum:	36
$0 \times 64 = 0$. Add 0 to sum:	36
$1 \times 128 = 128$. Add 128 to sum:	

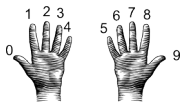
Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$. Add 0 to sum:	0
$1 \times 4 = 4$. Add 4 to sum:	4
$0 \times 8 = 0$. Add 0 to sum:	4
$0 \times 16 = 0$. Add 0 to sum:	4
$1 \times 32 = 32$. Add 32 to sum:	36
$0 \times 64 = 0$. Add 0 to sum:	36
$1 \times 128 = 128$. Add 128 to sum:	164

Recap: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:	0
$0 \times 2 = 0$. Add 0 to sum:	0
$1 \times 4 = 4$. Add 4 to sum:	4
$0 \times 8 = 0$. Add 0 to sum:	4
$0 \times 16 = 0$. Add 0 to sum:	4
$1 \times 32 = 32$. Add 32 to sum:	36
$0 \times 64 = 0$. Add 0 to sum:	36
$1 \times 128 = 128$. Add 128 to sum:	164

The answer is 164.

Today's Topics



- Data Representation
- **Machine Language: Jumps & Loops**
- Recap of Python & Circuits
- Design Patterns: Sorting

Recap: Machine Language

- We will be writing programs in a simplified machine language, WeMIPS.

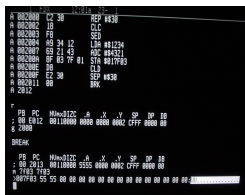
```

# 000000 C2 30 REP #430
# 000002 18 CLC
# 000004 FD JNC $+5
# 000006 #09 34 12 LMA #01224
# 000007 59 21 43 ADC #04321
# 000008 6F 03 7F 01 STA #01743
# 00000A 00 00 CLD
# 00000C 72 30 SEC #430
# 00000E 00 00 JNC $+5
# 000010 00 00
# 000012 00 00
# 000014 00 00
# 000016 00 00
# 000018 00 00
# 00001A 00 00
# 00001C 00 00
# 00001E 00 00
# 000020 00 00
# 000022 00 00
# 000024 00 00
# 000026 00 00
# 000028 00 00
# 00002A 00 00
# 00002C 00 00
# 00002E 00 00
# 000030 00 00
# 000032 00 00
# 000034 00 00
# 000036 00 00
# 000038 00 00
# 00003A 00 00
# 00003C 00 00
# 00003E 00 00
# 000040 00 00
# 000042 00 00
# 000044 00 00
# 000046 00 00
# 000048 00 00
# 00004A 00 00
# 00004C 00 00
# 00004E 00 00
# 000050 00 00
# 000052 00 00
# 000054 00 00
# 000056 00 00
# 000058 00 00
# 00005A 00 00
# 00005C 00 00
# 00005E 00 00
# 000060 00 00
# 000062 00 00
# 000064 00 00
# 000066 00 00
# 000068 00 00
# 00006A 00 00
# 00006C 00 00
# 00006E 00 00
# 000070 00 00
# 000072 00 00
# 000074 00 00
# 000076 00 00
# 000078 00 00
# 00007A 00 00
# 00007C 00 00
# 00007E 00 00
# 000080 00 00
# 000082 00 00
# 000084 00 00
# 000086 00 00
# 000088 00 00
# 00008A 00 00
# 00008C 00 00
# 00008E 00 00
# 000090 00 00
# 000092 00 00
# 000094 00 00
# 000096 00 00
# 000098 00 00
# 00009A 00 00
# 00009C 00 00
# 00009E 00 00
# 0000A0 00 00
# 0000A2 00 00
# 0000A4 00 00
# 0000A6 00 00
# 0000A8 00 00
# 0000AA 00 00
# 0000AC 00 00
# 0000AE 00 00
# 0000B0 00 00
# 0000B2 00 00
# 0000B4 00 00
# 0000B6 00 00
# 0000B8 00 00
# 0000BA 00 00
# 0000BC 00 00
# 0000BE 00 00
# 0000C0 00 00
# 0000C2 00 00
# 0000C4 00 00
# 0000C6 00 00
# 0000C8 00 00
# 0000CA 00 00
# 0000CC 00 00
# 0000CE 00 00
# 0000D0 00 00
# 0000D2 00 00
# 0000D4 00 00
# 0000D6 00 00
# 0000D8 00 00
# 0000DA 00 00
# 0000DC 00 00
# 0000DE 00 00
# 0000E0 00 00
# 0000E2 00 00
# 0000E4 00 00
# 0000E6 00 00
# 0000E8 00 00
# 0000EA 00 00
# 0000EC 00 00
# 0000EE 00 00
# 0000F0 00 00
# 0000F2 00 00
# 0000F4 00 00
# 0000F6 00 00
# 0000F8 00 00
# 0000FA 00 00
# 0000FC 00 00
# 0000FE 00 00
# 000100 00 00
# 000102 00 00
# 000104 00 00
# 000106 00 00
# 000108 00 00
# 00010A 00 00
# 00010C 00 00
# 00010E 00 00
# 000110 00 00
# 000112 00 00
# 000114 00 00
# 000116 00 00
# 000118 00 00
# 00011A 00 00
# 00011C 00 00
# 00011E 00 00
# 000120 00 00
# 000122 00 00
# 000124 00 00
# 000126 00 00
# 000128 00 00
# 00012A 00 00
# 00012C 00 00
# 00012E 00 00
# 000130 00 00
# 000132 00 00
# 000134 00 00
# 000136 00 00
# 000138 00 00
# 00013A 00 00
# 00013C 00 00
# 00013E 00 00
# 000140 00 00
# 000142 00 00
# 000144 00 00
# 000146 00 00
# 000148 00 00
# 00014A 00 00
# 00014C 00 00
# 00014E 00 00
# 000150 00 00
# 000152 00 00
# 000154 00 00
# 000156 00 00
# 000158 00 00
# 00015A 00 00
# 00015C 00 00
# 00015E 00 00
# 000160 00 00
# 000162 00 00
# 000164 00 00
# 000166 00 00
# 000168 00 00
# 00016A 00 00
# 00016C 00 00
# 00016E 00 00
# 000170 00 00
# 000172 00 00
# 000174 00 00
# 000176 00 00
# 000178 00 00
# 00017A 00 00
# 00017C 00 00
# 00017E 00 00
# 000180 00 00
# 000182 00 00
# 000184 00 00
# 000186 00 00
# 000188 00 00
# 00018A 00 00
# 00018C 00 00
# 00018E 00 00
# 000190 00 00
# 000192 00 00
# 000194 00 00
# 000196 00 00
# 000198 00 00
# 00019A 00 00
# 00019C 00 00
# 00019E 00 00
# 0001A0 00 00
# 0001A2 00 00
# 0001A4 00 00
# 0001A6 00 00
# 0001A8 00 00
# 0001AA 00 00
# 0001AC 00 00
# 0001AE 00 00
# 0001B0 00 00
# 0001B2 00 00
# 0001B4 00 00
# 0001B6 00 00
# 0001B8 00 00
# 0001BA 00 00
# 0001BC 00 00
# 0001BE 00 00
# 0001C0 00 00
# 0001C2 00 00
# 0001C4 00 00
# 0001C6 00 00
# 0001C8 00 00
# 0001CA 00 00
# 0001CC 00 00
# 0001CE 00 00
# 0001D0 00 00
# 0001D2 00 00
# 0001D4 00 00
# 0001D6 00 00
# 0001D8 00 00
# 0001DA 00 00
# 0001DC 00 00
# 0001DE 00 00
# 0001E0 00 00
# 0001E2 00 00
# 0001E4 00 00
# 0001E6 00 00
# 0001E8 00 00
# 0001EA 00 00
# 0001EC 00 00
# 0001EE 00 00
# 0001F0 00 00
# 0001F2 00 00
# 0001F4 00 00
# 0001F6 00 00
# 0001F8 00 00
# 0001FA 00 00
# 0001FC 00 00
# 0001FE 00 00
# 000200 00 00
# 000202 00 00
# 000204 00 00
# 000206 00 00
# 000208 00 00
# 00020A 00 00
# 00020C 00 00
# 00020E 00 00
# 000210 00 00
# 000212 00 00
# 000214 00 00
# 000216 00 00
# 000218 00 00
# 00021A 00 00
# 00021C 00 00

```

(wiki)

Recap: Machine Language



The screenshot shows a debugger window with two panels. The top panel displays assembly instructions with their addresses and hex values. The bottom panel shows the current state of the processor registers.

```
002000 c2 30 REP #30
002002 10 CLC
002003 f0 SED
002004 40 34 12 LDR #01204
002007 60 21 43 ROR #04321
002008 0f 03 7f 01 STA #017f03
00200c 00 CLD
00200f e2 30 SEP #30
002011 00 BRK
002012

PC PC Min32C A X Y SP BP BB
: 00 2012 00110000 0000 0000 0002 c7ff 0000 00
$ 2000

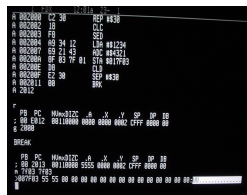
BREAK

PC PC Min32C A X Y SP BP BB
: 00 2013 00110000 5555 0000 0002 c7ff 0000 00
n 1103 7f03
007f03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(wiki)

- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.

Recap: Machine Language



```
002000 c2 30 REP #430
002002 10 CLC
002003 F0 SED
002004 40 34 12 LSH #1224
002007 60 21 43 RLC #4321
002008 0F 03 7F 01 STB #017F03
00200C 00 CLJ
00200F E2 30 SEP #430
002011 00 BRV
002012

PB PC Mem32C A X Y SP BP BB
: 00 2012 00110000 0000 0000 0002 C7FF 0000 00
$ 2000

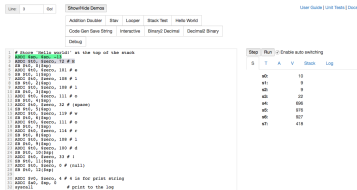
BREAK

PB PC Mem32C A X Y SP BP BB
: 00 2013 00110000 5555 0000 0002 C7FF 0000 00
n 1103 7403
007F03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(wiki)

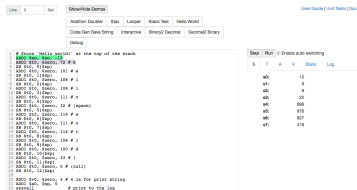
- We will be writing programs in a simplified machine language, WeMIPS.
- It is based on a reduced instruction set computer (RISC) design, originally developed by the MIPS Computer Systems.
- Due to its small set of commands, processors can be designed to run those commands very efficiently.

Recap: MIPS Commands



- **Registers:** locations for storing information that can be quickly accessed.

Recap: MIPS Commands



The screenshot shows a MIPS simulator window. On the left, there's a text area with assembly code. On the right, there's a 'Registers' window showing the current state of registers.

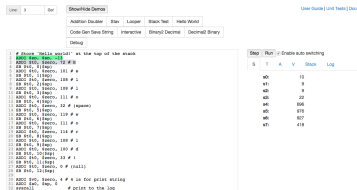
```
1 # Store "hello world" at the top of the stack
2 ADDI $0, $zero, 128
3 SW $0, 0($0)
4 ADDI $0, $zero, 191 # #
5 SW $0, 191($0)
6 ADDI $0, $zero, 100 # #
7 SW $0, 100($0)
8 ADDI $0, $zero, 111 # #
9 SW $0, 111($0)
10 ADDI $0, $zero, 12 # (space)
11 SW $0, 12($0)
12 ADDI $0, $zero, 11 # #
13 SW $0, 11($0)
14 ADDI $0, $zero, 114 # #
15 SW $0, 114($0)
16 ADDI $0, $zero, 100 # #
17 SW $0, 100($0)
18 ADDI $0, $zero, 33 # #
19 SW $0, 33($0)
20 ADDI $0, $zero, 0 # (null)
21 SW $0, 0($0)
22 ADDI $0, $zero, 4 # # is for print ending
23 ADDI $0, $zero, 0 # print to the log
24 syscall
```

The Registers window shows the following values:

Register	Value
\$0	10
\$1	9
\$2	9
\$3	22
\$4	856
\$5	878
\$6	877
\$7	418

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...

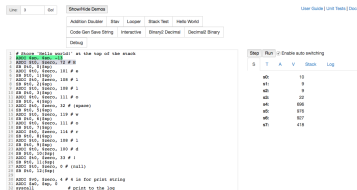
Recap: MIPS Commands



The screenshot shows a MIPS simulator interface. On the left, there's a text area with assembly code. On the right, there's a register window titled 'Registers' showing the values of registers \$0 through \$31.

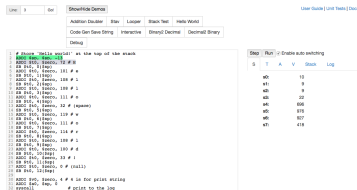
```
# Show "Hello world" as the top of the stack
1 ADDUI $0, $zero, 12 # 12
2 SW $0, 0($0)
3 ADDUI $0, $zero, 10 # 10
4 SW $0, 4($0)
5 ADDUI $0, $zero, 8 # 8
6 SW $0, 8($0)
7 ADDUI $0, $zero, 6 # 6
8 SW $0, 12($0)
9 ADDUI $0, $zero, 4 # 4
10 SW $0, 16($0)
11 ADDUI $0, $zero, 2 # 2
12 SW $0, 20($0)
13 ADDUI $0, $zero, 0 # 0
14 SW $0, 24($0)
15 ADDUI $0, $zero, 1 # 1
16 SW $0, 28($0)
17 ADDUI $0, $zero, 2 # 2
18 SW $0, 32($0)
19 ADDUI $0, $zero, 3 # 3
20 SW $0, 36($0)
21 ADDUI $0, $zero, 4 # 4
22 SW $0, 40($0)
23 ADDUI $0, $zero, 5 # 5
24 SW $0, 44($0)
25 ADDUI $0, $zero, 6 # 6
26 SW $0, 48($0)
27 ADDUI $0, $zero, 7 # 7
28 SW $0, 52($0)
29 ADDUI $0, $zero, 8 # 8
30 SW $0, 56($0)
31 ADDUI $0, $zero, 9 # 9
32 SW $0, 60($0)
33 ADDUI $0, $zero, 10 # 10
34 SW $0, 64($0)
35 ADDUI $0, $zero, 11 # 11
36 SW $0, 68($0)
37 ADDUI $0, $zero, 12 # 12
38 SW $0, 72($0)
39 ADDUI $0, $zero, 13 # 13
40 SW $0, 76($0)
41 ADDUI $0, $zero, 14 # 14
42 SW $0, 80($0)
43 ADDUI $0, $zero, 15 # 15
44 SW $0, 84($0)
45 ADDUI $0, $zero, 16 # 16
46 SW $0, 88($0)
47 ADDUI $0, $zero, 17 # 17
48 SW $0, 92($0)
49 ADDUI $0, $zero, 18 # 18
50 SW $0, 96($0)
51 ADDUI $0, $zero, 19 # 19
52 SW $0, 100($0)
53 ADDUI $0, $zero, 20 # 20
54 SW $0, 104($0)
55 ADDUI $0, $zero, 21 # 21
56 SW $0, 108($0)
57 ADDUI $0, $zero, 22 # 22
58 SW $0, 112($0)
59 ADDUI $0, $zero, 23 # 23
60 SW $0, 116($0)
61 ADDUI $0, $zero, 24 # 24
62 SW $0, 120($0)
63 ADDUI $0, $zero, 25 # 25
64 SW $0, 124($0)
65 ADDUI $0, $zero, 26 # 26
66 SW $0, 128($0)
67 ADDUI $0, $zero, 27 # 27
68 SW $0, 132($0)
69 ADDUI $0, $zero, 28 # 28
70 SW $0, 136($0)
71 ADDUI $0, $zero, 29 # 29
72 SW $0, 140($0)
73 ADDUI $0, $zero, 30 # 30
74 SW $0, 144($0)
75 ADDUI $0, $zero, 31 # 31
76 SW $0, 148($0)
77 ADDUI $0, $zero, 32 # 32
78 SW $0, 152($0)
79 ADDUI $0, $zero, 33 # 33
80 SW $0, 156($0)
81 ADDUI $0, $zero, 34 # 34
82 SW $0, 160($0)
83 ADDUI $0, $zero, 35 # 35
84 SW $0, 164($0)
85 ADDUI $0, $zero, 36 # 36
86 SW $0, 168($0)
87 ADDUI $0, $zero, 37 # 37
88 SW $0, 172($0)
89 ADDUI $0, $zero, 38 # 38
90 SW $0, 176($0)
91 ADDUI $0, $zero, 39 # 39
92 SW $0, 180($0)
93 ADDUI $0, $zero, 40 # 40
94 SW $0, 184($0)
95 ADDUI $0, $zero, 41 # 41
96 SW $0, 188($0)
97 ADDUI $0, $zero, 42 # 42
98 SW $0, 192($0)
99 ADDUI $0, $zero, 43 # 43
100 SW $0, 196($0)
101 ADDUI $0, $zero, 44 # 44
102 SW $0, 200($0)
103 ADDUI $0, $zero, 45 # 45
104 SW $0, 204($0)
105 ADDUI $0, $zero, 46 # 46
106 SW $0, 208($0)
107 ADDUI $0, $zero, 47 # 47
108 SW $0, 212($0)
109 ADDUI $0, $zero, 48 # 48
110 SW $0, 216($0)
111 ADDUI $0, $zero, 49 # 49
112 SW $0, 220($0)
113 ADDUI $0, $zero, 50 # 50
114 SW $0, 224($0)
115 ADDUI $0, $zero, 51 # 51
116 SW $0, 228($0)
117 ADDUI $0, $zero, 52 # 52
118 SW $0, 232($0)
119 ADDUI $0, $zero, 53 # 53
120 SW $0, 236($0)
121 ADDUI $0, $zero, 54 # 54
122 SW $0, 240($0)
123 ADDUI $0, $zero, 55 # 55
124 SW $0, 244($0)
125 ADDUI $0, $zero, 56 # 56
126 SW $0, 248($0)
127 ADDUI $0, $zero, 57 # 57
128 SW $0, 252($0)
129 ADDUI $0, $zero, 58 # 58
130 SW $0, 256($0)
131 ADDUI $0, $zero, 59 # 59
132 SW $0, 260($0)
133 ADDUI $0, $zero, 60 # 60
134 SW $0, 264($0)
135 ADDUI $0, $zero, 61 # 61
136 SW $0, 268($0)
137 ADDUI $0, $zero, 62 # 62
138 SW $0, 272($0)
139 ADDUI $0, $zero, 63 # 63
140 SW $0, 276($0)
141 ADDUI $0, $zero, 64 # 64
142 SW $0, 280($0)
143 ADDUI $0, $zero, 65 # 65
144 SW $0, 284($0)
145 ADDUI $0, $zero, 66 # 66
146 SW $0, 288($0)
147 ADDUI $0, $zero, 67 # 67
148 SW $0, 292($0)
149 ADDUI $0, $zero, 68 # 68
150 SW $0, 296($0)
151 ADDUI $0, $zero, 69 # 69
152 SW $0, 300($0)
153 ADDUI $0, $zero, 70 # 70
154 SW $0, 304($0)
155 ADDUI $0, $zero, 71 # 71
156 SW $0, 308($0)
157 ADDUI $0, $zero, 72 # 72
158 SW $0, 312($0)
159 ADDUI $0, $zero, 73 # 73
160 SW $0, 316($0)
161 ADDUI $0, $zero, 74 # 74
162 SW $0, 320($0)
163 ADDUI $0, $zero, 75 # 75
164 SW $0, 324($0)
165 ADDUI $0, $zero, 76 # 76
166 SW $0, 328($0)
167 ADDUI $0, $zero, 77 # 77
168 SW $0, 332($0)
169 ADDUI $0, $zero, 78 # 78
170 SW $0, 336($0)
171 ADDUI $0, $zero, 79 # 79
172 SW $0, 340($0)
173 ADDUI $0, $zero, 80 # 80
174 SW $0, 344($0)
175 ADDUI $0, $zero, 81 # 81
176 SW $0, 348($0)
177 ADDUI $0, $zero, 82 # 82
178 SW $0, 352($0)
179 ADDUI $0, $zero, 83 # 83
180 SW $0, 356($0)
181 ADDUI $0, $zero, 84 # 84
182 SW $0, 360($0)
183 ADDUI $0, $zero, 85 # 85
184 SW $0, 364($0)
185 ADDUI $0, $zero, 86 # 86
186 SW $0, 368($0)
187 ADDUI $0, $zero, 87 # 87
188 SW $0, 372($0)
189 ADDUI $0, $zero, 88 # 88
190 SW $0, 376($0)
191 ADDUI $0, $zero, 89 # 89
192 SW $0, 380($0)
193 ADDUI $0, $zero, 90 # 90
194 SW $0, 384($0)
195 ADDUI $0, $zero, 91 # 91
196 SW $0, 388($0)
197 ADDUI $0, $zero, 92 # 92
198 SW $0, 392($0)
199 ADDUI $0, $zero, 93 # 93
200 SW $0, 396($0)
201 ADDUI $0, $zero, 94 # 94
202 SW $0, 400($0)
203 ADDUI $0, $zero, 95 # 95
204 SW $0, 404($0)
205 ADDUI $0, $zero, 96 # 96
206 SW $0, 408($0)
207 ADDUI $0, $zero, 97 # 97
208 SW $0, 412($0)
209 ADDUI $0, $zero, 98 # 98
210 SW $0, 416($0)
211 ADDUI $0, $zero, 99 # 99
212 SW $0, 420($0)
213 ADDUI $0, $zero, 100 # 100
214 SW $0, 424($0)
215 ADDUI $0, $zero, 101 # 101
216 SW $0, 428($0)
217 ADDUI $0, $zero, 102 # 102
218 SW $0, 432($0)
219 ADDUI $0, $zero, 103 # 103
220 SW $0, 436($0)
221 ADDUI $0, $zero, 104 # 104
222 SW $0, 440($0)
223 ADDUI $0, $zero, 105 # 105
224 SW $0, 444($0)
225 ADDUI $0, $zero, 106 # 106
226 SW $0, 448($0)
227 ADDUI $0, $zero, 107 # 107
228 SW $0, 452($0)
229 ADDUI $0, $zero, 108 # 108
230 SW $0, 456($0)
231 ADDUI $0, $zero, 109 # 109
232 SW $0, 460($0)
233 ADDUI $0, $zero, 110 # 110
234 SW $0, 464($0)
235 ADDUI $0, $zero, 111 # 111
236 SW $0, 468($0)
237 ADDUI $0, $zero, 112 # 112
238 SW $0, 472($0)
239 ADDUI $0, $zero, 113 # 113
240 SW $0, 476($0)
241 ADDUI $0, $zero, 114 # 114
242 SW $0, 480($0)
243 ADDUI $0, $zero, 115 # 115
244 SW $0, 484($0)
245 ADDUI $0, $zero, 116 # 116
246 SW $0, 488($0)
247 ADDUI $0, $zero, 117 # 117
248 SW $0, 492($0)
249 ADDUI $0, $zero, 118 # 118
250 SW $0, 496($0)
251 ADDUI $0, $zero, 119 # 119
252 SW $0, 500($0)
253 ADDUI $0, $zero, 120 # 120
254 SW $0, 504($0)
255 ADDUI $0, $zero, 121 # 121
256 SW $0, 508($0)
257 ADDUI $0, $zero, 122 # 122
258 SW $0, 512($0)
259 ADDUI $0, $zero, 123 # 123
260 SW $0, 516($0)
261 ADDUI $0, $zero, 124 # 124
262 SW $0, 520($0)
263 ADDUI $0, $zero, 125 # 125
264 SW $0, 524($0)
265 ADDUI $0, $zero, 126 # 126
266 SW $0, 528($0)
267 ADDUI $0, $zero, 127 # 127
268 SW $0, 532($0)
269 ADDUI $0, $zero, 128 # 128
270 SW $0, 536($0)
271 ADDUI $0, $zero, 129 # 129
272 SW $0, 540($0)
273 ADDUI $0, $zero, 130 # 130
274 SW $0, 544($0)
275 ADDUI $0, $zero, 131 # 131
276 SW $0, 548($0)
277 ADDUI $0, $zero, 132 # 132
278 SW $0, 552($0)
279 ADDUI $0, $zero, 133 # 133
280 SW $0, 556($0)
281 ADDUI $0, $zero, 134 # 134
282 SW $0, 560($0)
283 ADDUI $0, $zero, 135 # 135
284 SW $0, 564($0)
285 ADDUI $0, $zero, 136 # 136
286 SW $0, 568($0)
287 ADDUI $0, $zero, 137 # 137
288 SW $0, 572($0)
289 ADDUI $0, $zero, 138 # 138
290 SW $0, 576($0)
291 ADDUI $0, $zero, 139 # 139
292 SW $0, 580($0)
293 ADDUI $0, $zero, 140 # 140
294 SW $0, 584($0)
295 ADDUI $0, $zero, 141 # 141
296 SW $0, 588($0)
297 ADDUI $0, $zero, 142 # 142
298 SW $0, 592($0)
299 ADDUI $0, $zero, 143 # 143
300 SW $0, 596($0)
301 ADDUI $0, $zero, 144 # 144
302 SW $0, 600($0)
303 ADDUI $0, $zero, 145 # 145
304 SW $0, 604($0)
305 ADDUI $0, $zero, 146 # 146
306 SW $0, 608($0)
307 ADDUI $0, $zero, 147 # 147
308 SW $0, 612($0)
309 ADDUI $0, $zero, 148 # 148
310 SW $0, 616($0)
311 ADDUI $0, $zero, 149 # 149
312 SW $0, 620($0)
313 ADDUI $0, $zero, 150 # 150
314 SW $0, 624($0)
315 ADDUI $0, $zero, 151 # 151
316 SW $0, 628($0)
317 ADDUI $0, $zero, 152 # 152
318 SW $0, 632($0)
319 ADDUI $0, $zero, 153 # 153
320 SW $0, 636($0)
321 ADDUI $0, $zero, 154 # 154
322 SW $0, 640($0)
323 ADDUI $0, $zero, 155 # 155
324 SW $0, 644($0)
325 ADDUI $0, $zero, 156 # 156
326 SW $0, 648($0)
327 ADDUI $0, $zero, 157 # 157
328 SW $0, 652($0)
329 ADDUI $0, $zero, 158 # 158
330 SW $0, 656($0)
331 ADDUI $0, $zero, 159 # 159
332 SW $0, 660($0)
333 ADDUI $0, $zero, 160 # 160
334 SW $0, 664($0)
335 ADDUI $0, $zero, 161 # 161
336 SW $0, 668($0)
337 ADDUI $0, $zero, 162 # 162
338 SW $0, 672($0)
339 ADDUI $0, $zero, 163 # 163
340 SW $0, 676($0)
341 ADDUI $0, $zero, 164 # 164
342 SW $0, 680($0)
343 ADDUI $0, $zero, 165 # 165
344 SW $0, 684($0)
345 ADDUI $0, $zero, 166 # 166
346 SW $0, 688($0)
347 ADDUI $0, $zero, 167 # 167
348 SW $0, 692($0)
349 ADDUI $0, $zero, 168 # 168
350 SW $0, 696($0)
351 ADDUI $0, $zero, 169 # 169
352 SW $0, 700($0)
353 ADDUI $0, $zero, 170 # 170
354 SW $0, 704($0)
355 ADDUI $0, $zero, 171 # 171
356 SW $0, 708($0)
357 ADDUI $0, $zero, 172 # 172
358 SW $0, 712($0)
359 ADDUI $0, $zero, 173 # 173
360 SW $0, 716($0)
361 ADDUI $0, $zero, 174 # 174
362 SW $0, 720($0)
363 ADDUI $0, $zero, 175 # 175
364 SW $0, 724($0)
365 ADDUI $0, $zero, 176 # 176
366 SW $0, 728($0)
367 ADDUI $0, $zero, 177 # 177
368 SW $0, 732($0)
369 ADDUI $0, $zero, 178 # 178
370 SW $0, 736($0)
371 ADDUI $0, $zero, 179 # 179
372 SW $0, 740($0)
373 ADDUI $0, $zero, 180 # 180
374 SW $0, 744($0)
375 ADDUI $0, $zero, 181 # 181
376 SW $0, 748($0)
377 ADDUI $0, $zero, 182 # 182
378 SW $0, 752($0)
379 ADDUI $0, $zero, 183 # 183
380 SW $0, 756($0)
381 ADDUI $0, $zero, 184 # 184
382 SW $0, 760($0)
383 ADDUI $0, $zero, 185 # 185
384 SW $0, 764($0)
385 ADDUI $0, $zero, 186 # 186
386 SW $0, 768($0)
387 ADDUI $0, $zero, 187 # 187
388 SW $0, 772($0)
389 ADDUI $0, $zero, 188 # 188
390 SW $0, 776($0)
391 ADDUI $0, $zero, 189 # 189
392 SW $0, 780($0)
393 ADDUI $0, $zero, 190 # 190
394 SW $0, 784($0)
395 ADDUI $0, $zero, 191 # 191
396 SW $0, 788($0)
397 ADDUI $0, $zero, 192 # 192
398 SW $0, 792($0)
399 ADDUI $0, $zero, 193 # 193
400 SW $0, 796($0)
401 ADDUI $0, $zero, 194 # 194
402 SW $0, 800($0)
403 ADDUI $0, $zero, 195 # 195
404 SW $0, 804($0)
405 ADDUI $0, $zero, 196 # 196
406 SW $0, 808($0)
407 ADDUI $0, $zero, 197 # 197
408 SW $0, 812($0)
409 ADDUI $0, $zero, 198 # 198
410 SW $0, 816($0)
411 ADDUI $0, $zero, 199 # 199
412 SW $0, 820($0)
413 ADDUI $0, $zero, 200 # 200
414 SW $0, 824($0)
415 ADDUI $0, $zero, 201 # 201
416 SW $0, 828($0)
417 ADDUI $0, $zero, 202 # 202
418 SW $0, 832($0)
419 ADDUI $0, $zero, 203 # 203
420 SW $0, 836($0)
421 ADDUI $0, $zero, 204 # 204
422 SW $0, 840($0)
423 ADDUI $0, $zero, 205 # 205
424 SW $0, 844($0)
425 ADDUI $0, $zero, 206 # 206
426 SW $0, 848($0)
427 ADDUI $0, $zero, 207 # 207
428 SW $0, 852($0)
429 ADDUI $0, $zero, 208 # 208
430 SW $0, 856($0)
431 ADDUI $0, $zero, 209 # 209
432 SW $0, 860($0)
433 ADDUI $0, $zero, 210 # 210
434 SW $0, 864($0)
435 ADDUI $0, $zero, 211 # 211
436 SW $0, 868($0)
437 ADDUI $0, $zero, 212 # 212
438 SW $0, 872($0)
439 ADDUI $0, $zero, 213 # 213
440 SW $0, 876($0)
441 ADDUI $0, $zero, 214 # 214
442 SW $0, 880($0)
443 ADDUI $0, $zero, 215 # 215
444 SW $0, 884($0)
445 ADDUI $0, $zero, 216 # 216
446 SW $0, 888($0)
447 ADDUI $0, $zero, 217 # 217
448 SW $0, 892($0)
449 ADDUI $0, $zero, 218 # 218
450 SW $0, 896($0)
451 ADDUI $0, $zero, 219 # 219
452 SW $0, 900($0)
453 ADDUI $0, $zero, 220 # 220
454 SW $0, 904($0)
455 ADDUI $0, $zero, 221 # 221
456 SW $0, 908($0)
457 ADDUI $0, $zero, 222 # 222
458 SW $0, 912($0)
459 ADDUI $0, $zero, 223 # 223
460 SW $0, 916($0)
461 ADDUI $0, $zero, 224 # 224
462 SW $0, 920($0)
463 ADDUI $0, $zero, 225 # 225
464 SW $0, 924($0)
465 ADDUI $0, $zero, 226 # 226
466 SW $0, 928($0)
467 ADDUI $0, $zero, 227 # 227
468 SW $0, 932($0)
469 ADDUI $0, $zero, 228 # 228
470 SW $0, 936($0)
471 ADDUI $0, $zero, 229 # 229
472 SW $0, 940($0)
473 ADDUI $0, $zero, 230 # 230
474 SW $0, 944($0)
475 ADDUI $0, $zero, 231 # 231
476 SW $0, 948($0)
477 ADDUI $0, $zero, 232 # 232
478 SW $0, 952($0)
479 ADDUI $0, $zero, 233 # 233
480 SW $0, 956($0)
481 ADDUI $0, $zero, 234 # 234
482 SW $0, 960($0)
483 ADDUI $0, $zero, 235 # 235
484 SW $0, 964($0)
485 ADDUI $0, $zero, 236 # 236
486 SW $0, 968($0)
487 ADDUI $0, $zero, 237 # 237
488 SW $0, 972($0)
489 ADDUI $0, $zero, 238 # 238
490 SW $0, 976($0)
491 ADDUI $0, $zero, 239 # 239
492 SW $0, 980($0)
493 ADDUI $0, $zero, 240 # 240
494 SW $0, 984($0)
495 ADDUI $0, $zero, 241 # 241
496 SW $0, 988($0)
497 ADDUI $0, $zero, 242 # 242
498 SW $0, 992($0)
499 ADDUI $0, $zero, 243 # 243
500 SW $0, 996($0)
501 ADDUI $0, $zero, 244 # 244
502 SW $0, 1000($0)
503 ADDUI $0, $zero, 245 # 245
504 SW $0, 1004($0)
505 ADDUI $0, $zero, 246 # 246
506 SW $0, 1008($0)
507 ADDUI $0, $zero, 247 # 247
508 SW $0, 1012($0)
509 ADDUI $0, $zero, 248 # 248
510 SW $0, 1016($0)
511 ADDUI $0, $zero, 249 # 249
512 SW $0, 1020($0)
513 ADDUI $0, $zero, 250 # 250
514 SW $0, 1024($0)
515 ADDUI $0, $zero, 251 # 251
516 SW $0, 1028($0)
517 ADDUI $0, $zero, 252 # 252
518 SW $0, 1032($0)
519 ADDUI $0, $zero, 253 # 253
520 SW $0, 1036($0)
521 ADDUI $0, $zero, 254 # 254
522 SW $0, 1040($0)
523 ADDUI $0, $zero, 255 # 255
524 SW $0, 1044($0)
525 ADDUI $0, $zero, 256 # 256
526 SW $0, 1048($0)
527 ADDUI $0, $zero, 257 # 257
528 SW $0, 1052($0)
529 ADDUI $0, $zero, 258 # 258
530 SW $0, 1056($0)
531 ADDUI $0, $zero, 259 # 259
532 SW $0, 1060($0)
533 ADDUI $0, $zero, 260 # 260
534 SW $0, 1064($0)
535 ADDUI $0, $zero, 261 # 261
536 SW $0, 1068($0)
537 ADDUI $0, $zero, 262 # 262
538 SW $0, 1072($0)
539 ADDUI $0, $zero, 263 # 263
540 SW $0, 1076($0)
541 ADDUI $0, $zero, 264 # 264
542 SW $0, 1080($0)
543 ADDUI $0, $zero, 265 # 265
544 SW $0, 1084($0)
545 ADDUI $0, $zero, 266 # 266
546 SW $0, 1088($0)
547 ADDUI $0, $zero, 267 # 267
548 SW $0, 1092($0)
549 ADDUI $0, $zero, 268 # 268
550 SW $0, 1096($0)
551 ADDUI $0, $zero, 269 # 269
552 SW $0, 1100($0)
553 ADDUI $0, $zero, 270 # 270
554 SW $0, 1104($0)
555 ADDUI $0, $zero, 271 # 271
556 SW $0, 1108($0)
557 ADDUI $0, $zero, 272 # 272
558 SW $0, 1112($0)
559 ADDUI $0, $zero, 273 # 273
560 SW $0, 1116($0)
561 ADDUI $0, $zero, 274 # 274
562 SW $0, 1120($0)
563 ADDUI $0, $zero, 275 # 275
564 SW $0, 1124($0)
565 ADDUI $0, $zero, 276 # 276
566 SW $0, 1128($0)
567 ADDUI $0, $zero, 277 # 277
568 SW $0, 1132($0)
569 ADDUI $0, $zero, 278 # 278
570 SW $0, 1136($0)
571 ADDUI $0, $zero, 279 # 279
572 SW $0, 1140($0)
573 ADDUI $0, $zero, 280 # 280
574 SW $0, 1144($0)
575 ADDUI $0, $zero, 281 # 281
576 SW $0, 1148($0)
577 ADDUI $0, $zero, 282 # 282
578 SW $0, 1152($0)
579 ADDUI $0, $zero, 283 # 283
580 SW $0, 1156($0)
581 ADDUI $0, $zero, 284 # 284
582 SW $0, 1160($0)
583 ADDUI $0, $zero, 285 # 285
584 SW $0, 1164($0)
585 ADDUI $0, $zero, 286 # 286
586 SW $0, 1168($0)
587 ADDUI $0, $zero, 287 # 287
588 SW $0, 1172($0)
589 ADDUI $0, $zero, 288 # 288
590 SW $0, 1176($0)
591 ADDUI $0, $zero, 289 # 289
592 SW $0, 1180($0)
593 ADDUI $0, $zero, 290 # 290
594 SW $0, 1184($0)
595 ADDUI $0, $zero, 291 # 291
596 SW $0, 1188($0)
597 ADDUI $0, $zero, 292 # 292
598 SW $0, 1192($0)
599 ADDUI $0, $zero, 293 # 293
600 SW $0, 1196($0)
601 ADDUI $0, $zero, 294 # 294
602 SW $0, 1200($0)
603 ADDUI $0, $zero, 295 # 295
604 SW $0, 1204($0)
605 ADDUI $0, $zero, 296 # 296
606 SW $0, 1208($0)
607 ADDUI $0, $zero, 297 # 297
608 SW $0, 1212($0)
609 ADDUI $0, $zero, 298 # 298
610 SW $0, 1216($0)
611 ADDUI $0, $zero, 299 # 299
612 SW $0, 1220($0)
613 ADDUI $0, $zero, 300 # 300
614 SW $0, 1224($0)
615 ADDUI $0, $zero, 301 # 301
616 SW $0, 1228($0)
617 ADDUI $0, $zero, 302 # 302
618 SW $0, 1232($0)
619 ADDUI $0, $zero, 303 # 303
620 SW $0, 1236($0)
621 ADDUI $0, $zero, 304 # 304
622 SW $0, 1240($0)
623 ADDUI $0, $zero, 305 # 305
624 SW $0, 1244($0)
625 ADDUI $0, $zero, 306 # 306
626 SW $0, 1248($0)
627 ADDUI $0, $zero, 307 # 307
628 SW $0, 1252($0)
629 ADDUI $0, $zero, 308 # 308
630 SW $0, 1256($0)
631 ADDUI $0, $zero, 309 # 309
632 SW $0, 1260($0)
633 ADDUI $0, $zero, 310 # 310
634 SW $0, 1264($0)
635 ADDUI $0, $zero, 311 # 311
636 SW $0, 1268($0)
637 ADDUI $0, $zero, 312 # 312
638 SW $0, 1272($0)
639 ADDUI $0, $zero, 313 # 313
640 SW $0, 1276($0)
641 ADDUI $0, $zero, 314 # 314
642 SW $0, 1280($0)
643 ADDUI $0, $zero, 315 # 315
644 SW $0, 1284($0)
645 ADDUI $0, $zero, 316 # 316
646 SW $0, 1288($0)
647 ADDUI $0, $zero, 317 # 317
648 SW $0, 1292($0)
649 ADDUI $0, $zero, 318 # 318
650 SW $0, 1296($0)
651 ADDUI $0, $zero, 319 # 319
652 SW $0, 1300($0)
653 ADDUI $0, $zero, 320 # 320
654 SW $0, 1304($0)
655 ADDUI $0, $zero, 321 # 321
656 SW $0, 1308($0)
657 ADDUI $0, $zero, 322 # 322
658 SW $0, 1312($0)
659 ADDUI $0, $zero, 323 # 323
660 SW $0, 1316($0)
661 ADDUI $0, $zero, 324 # 324
662 SW $0, 1320($0)
663 ADDUI $0, $zero, 325 # 325
664 SW $0, 1324($0)
665 ADDUI $0, $zero, 326 # 326
666 SW $0, 1328($0)
667 ADDUI $0, $zero, 327 # 327
668 SW $0, 1332($0)
669 ADDUI $0, $zero, 328 # 328
670 SW $0, 1336($0)
671 ADDUI $0, $zero, 329 # 329
672 SW $0, 1340($0)
673 ADDUI $0, $zero, 330 # 330
674 SW $0, 1344($0)
675 ADDUI $0, $zero, 331 # 331
676 SW $0, 1348($0)
677 ADDUI $0, $zero, 332 # 332
678 SW $0, 1352($0)
679 ADDUI $0, $zero, 333 # 333
680 SW $0, 1356($0)
681 ADDUI $0, $zero, 334 # 334
682 SW $0, 1360($0)
683 ADDUI $0, $zero, 335 # 335
684 SW $0, 1364($0)
685 ADDUI $0, $zero, 336 # 336
686 SW $0, 1368($0)
687 ADDUI $0, $zero, 337 # 337
688 SW $0, 1372($0)
689 ADDUI $0, $zero, 338 # 338
690 SW $0, 1376($0)
691 ADDUI $0, $zero, 339 # 339
692 SW $0, 1380($0)
693 ADDUI $0, $zero, 340 # 340
694 SW $0, 1384($0)
695 ADD
```

Recap: MIPS Commands



- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:
add \$s1, \$s2, \$s3

Recap: MIPS Commands



The screenshot shows a MIPS simulator window. On the left, there's a text area with assembly code. On the right, there's a 'Registers' window showing the state of various registers.

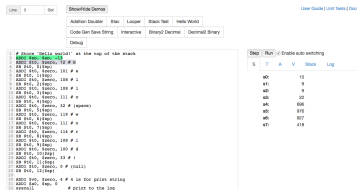
```
1 # Choose "Hello World" as the top of the stack
2 ADDUI $t0, $zero, 12 # 12
3 SW $t0, 0($zero)
4 ADDUI $t1, $zero, 181 # 181
5 SW $t1, 4($zero)
6 ADDUI $t2, $zero, 100 # 100
7 SW $t2, 8($zero)
8 ADDUI $t3, $zero, 100 # 100
9 SW $t3, 12($zero)
10 ADDUI $t4, $zero, 111 # 111
11 SW $t4, 16($zero)
12 ADDUI $t5, $zero, 12 # (space)
13 SW $t5, 20($zero)
14 ADDUI $t6, $zero, 110 # 110
15 SW $t6, 24($zero)
16 ADDUI $t7, $zero, 111 # 111
17 SW $t7, 28($zero)
18 ADDUI $t8, $zero, 114 # 114
19 SW $t8, 32($zero)
20 ADDUI $t9, $zero, 100 # 100
21 SW $t9, 36($zero)
22 ADDUI $t10, $zero, 100 # 100
23 SW $t10, 40($zero)
24 ADDUI $t11, $zero, 10 # 10
25 SW $t11, 44($zero)
26 ADDUI $t12, $zero, 0 # (null)
27 SW $t12, 48($zero)
28 ADDUI $t13, $zero, 4 # 4 is for print ending
29 ADDUI $t14, $zero, 0
30 syscall # print to the log
```

The Registers window shows the following values:

Register	Value
\$t0	12
\$t1	181
\$t2	100
\$t3	100
\$t4	111
\$t5	12
\$t6	110
\$t7	111
\$t8	114
\$t9	100
\$t10	100
\$t11	10
\$t12	0
\$t13	4
\$t14	0

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:
add \$s1, \$s2, \$s3 (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.

Recap: MIPS Commands



The screenshot shows a MIPS simulator window. On the left, there's a text area with assembly code. On the right, there's a register window showing the state of registers \$0 through \$31.

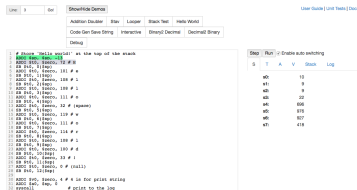
```
1: # Choose "Hello world" as the top of the stack
2: ADDI $0, $zero, 12 # R
3: SW $0, 12($0)
4: ADDI $0, $zero, 181 # R
5: SW $0, 181($0)
6: ADDI $0, $zero, 100 # I
7: SW $0, 100($0)
8: ADDI $0, $zero, 100 # I
9: SW $0, 100($0)
10: ADDI $0, $zero, 111 # R
11: SW $0, 111($0)
12: ADDI $0, $zero, 12 # (again)
13: SW $0, 12($0)
14: ADDI $0, $zero, 111 # R
15: SW $0, 111($0)
16: ADDI $0, $zero, 114 # R
17: SW $0, 114($0)
18: ADDI $0, $zero, 100 # I
19: SW $0, 100($0)
20: ADDI $0, $zero, 100 # R
21: SW $0, 100($0)
22: ADDI $0, $zero, 10 # I
23: SW $0, 10($0)
24: ADDI $0, $zero, 0 # (null)
25: SW $0, 0($0)
26: ADDI $0, $zero, 4 # 4 is for print ending
27: ADDI $0, $zero, 4
28: syscall # print to the log
```

The register window on the right shows the following values:

Register	Value
\$0	10
\$1	9
\$2	9
\$3	22
\$4	100
\$5	100
\$6	111
\$7	111
\$8	114
\$9	100
\$10	100
\$11	10
\$12	0
\$13	4
\$14	4
\$15	4
\$16	4
\$17	4
\$18	4
\$19	4
\$20	4
\$21	4
\$22	4
\$23	4
\$24	4
\$25	4
\$26	4
\$27	4
\$28	4
\$29	4
\$30	4
\$31	4

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:
add \$s1, \$s2, \$s3 (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.
addi \$s1, \$s2, 100

Recap: MIPS Commands



```
1 # Please "hello world" as the top of the main
2 addi $v0, $zero, 12 # 12
3 li $t0, 10
4 add $a0, $zero, 10 # 10
5 li $t1, 10000
6 add $t2, $zero, 100 # 100
7 li $t3, 10000
8 add $t4, $zero, 100 # 100
9 li $t5, 10000
10 add $t6, $zero, 111 # 111
11 li $t7, 10000
12 add $t8, $zero, 12 # (space)
13 li $t9, 10000
14 add $t10, $zero, 111 # 111
15 li $t11, 10000
16 add $t12, $zero, 114 # 114
17 li $t13, 10000
18 add $t14, $zero, 100 # 100
19 li $t15, 10000
20 add $t16, $zero, 33 # 33
21 li $t17, 10000
22 add $t18, $zero, 0 # (null)
23 li $t19, 10000
24 add $t20, $zero, 4 # 4 is for print string
25 add $t21, $zero, 6
26 syscall
27 # print to the log
```

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:
add \$s1, \$s2, \$s3 (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.
addi \$s1, \$s2, 100 (Basic form: OP rd, rs, imm)
- **J Instructions:** instructions that jump to another memory location.

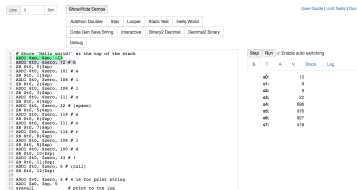
Recap: MIPS Commands

```

1  # Store "Hello world" at the top of the stack
2  ADDI $sp, $zero, 20 #R
3  # R10, 1000
4  ADDI $t0, $zero, 181 #R
5  # R10, 1000
6  ADDI $t0, $zero, 188 # R
7  # R10, 1000
8  ADDI $t0, $zero, 189 # R
9  # R10, 1000
10 ADDI $t0, $zero, 191 # R
11 # R10, 1000
12 ADDI $t0, $zero, 192 # (page)
13 # R10, 1000
14 ADDI $t0, $zero, 193 # R
15 # R10, 1000
16 ADDI $t0, $zero, 194 # R
17 # R10, 1000
18 ADDI $t0, $zero, 195 # R
19 # R10, 1000
20 ADDI $t0, $zero, 196 # R
21 # R10, 1000
22 ADDI $t0, $zero, 197 # R
23 # R10, 1000
24 ADDI $t0, $zero, 198 # R
25 # R10, 1000
26 ADDI $t0, $zero, 199 # R
27 # R10, 1000
28 ADDI $t0, $zero, 200 # R
29 # R10, 1000
30 ADDI $t0, $zero, 201 # R
31 # R10, 1000
32 ADDI $t0, $zero, 202 # R
33 # R10, 1000
34 ADDI $t0, $zero, 203 # R
35 # R10, 1000
36 ADDI $t0, $zero, 204 # R
37 # R10, 1000
38 ADDI $t0, $zero, 205 # R
39 # R10, 1000
40 ADDI $t0, $zero, 206 # R
41 # R10, 1000
42 ADDI $t0, $zero, 207 # R
43 # R10, 1000
44 ADDI $t0, $zero, 208 # R
45 # R10, 1000
46 ADDI $t0, $zero, 209 # R
47 # R10, 1000
48 ADDI $t0, $zero, 210 # R
49 # R10, 1000
50 ADDI $t0, $zero, 211 # R
51 # R10, 1000
52 ADDI $t0, $zero, 212 # R
53 # R10, 1000
54 ADDI $t0, $zero, 213 # R
55 # R10, 1000
56 ADDI $t0, $zero, 214 # R
57 # R10, 1000
58 ADDI $t0, $zero, 215 # R
59 # R10, 1000
60 ADDI $t0, $zero, 216 # R
61 # R10, 1000
62 ADDI $t0, $zero, 217 # R
63 # R10, 1000
64 ADDI $t0, $zero, 218 # R
65 # R10, 1000
66 ADDI $t0, $zero, 219 # R
67 # R10, 1000
68 ADDI $t0, $zero, 220 # R
69 # R10, 1000
70 ADDI $t0, $zero, 221 # R
71 # R10, 1000
72 ADDI $t0, $zero, 222 # R
73 # R10, 1000
74 ADDI $t0, $zero, 223 # R
75 # R10, 1000
76 ADDI $t0, $zero, 224 # R
77 # R10, 1000
78 ADDI $t0, $zero, 225 # R
79 # R10, 1000
80 ADDI $t0, $zero, 226 # R
81 # R10, 1000
82 ADDI $t0, $zero, 227 # R
83 # R10, 1000
84 ADDI $t0, $zero, 228 # R
85 # R10, 1000
86 ADDI $t0, $zero, 229 # R
87 # R10, 1000
88 ADDI $t0, $zero, 230 # R
89 # R10, 1000
90 ADDI $t0, $zero, 231 # R
91 # R10, 1000
92 ADDI $t0, $zero, 232 # R
93 # R10, 1000
94 ADDI $t0, $zero, 233 # R
95 # R10, 1000
96 ADDI $t0, $zero, 234 # R
97 # R10, 1000
98 ADDI $t0, $zero, 235 # R
99 # R10, 1000
100 ADDI $t0, $zero, 236 # R
101 # R10, 1000
102 ADDI $t0, $zero, 237 # R
103 # R10, 1000
104 ADDI $t0, $zero, 238 # R
105 # R10, 1000
106 ADDI $t0, $zero, 239 # R
107 # R10, 1000
108 ADDI $t0, $zero, 240 # R
109 # R10, 1000
110 ADDI $t0, $zero, 241 # R
111 # R10, 1000
112 ADDI $t0, $zero, 242 # R
113 # R10, 1000
114 ADDI $t0, $zero, 243 # R
115 # R10, 1000
116 ADDI $t0, $zero, 244 # R
117 # R10, 1000
118 ADDI $t0, $zero, 245 # R
119 # R10, 1000
120 ADDI $t0, $zero, 246 # R
121 # R10, 1000
122 ADDI $t0, $zero, 247 # R
123 # R10, 1000
124 ADDI $t0, $zero, 248 # R
125 # R10, 1000
126 ADDI $t0, $zero, 249 # R
127 # R10, 1000
128 ADDI $t0, $zero, 250 # R
129 # R10, 1000
130 ADDI $t0, $zero, 251 # R
131 # R10, 1000
132 ADDI $t0, $zero, 252 # R
133 # R10, 1000
134 ADDI $t0, $zero, 253 # R
135 # R10, 1000
136 ADDI $t0, $zero, 254 # R
137 # R10, 1000
138 ADDI $t0, $zero, 255 # R
139 # R10, 1000
140 ADDI $t0, $zero, 256 # R
141 # R10, 1000
142 ADDI $t0, $zero, 257 # R
143 # R10, 1000
144 ADDI $t0, $zero, 258 # R
145 # R10, 1000
146 ADDI $t0, $zero, 259 # R
147 # R10, 1000
148 ADDI $t0, $zero, 260 # R
149 # R10, 1000
150 ADDI $t0, $zero, 261 # R
151 # R10, 1000
152 ADDI $t0, $zero, 262 # R
153 # R10, 1000
154 ADDI $t0, $zero, 263 # R
155 # R10, 1000
156 ADDI $t0, $zero, 264 # R
157 # R10, 1000
158 ADDI $t0, $zero, 265 # R
159 # R10, 1000
160 ADDI $t0, $zero, 266 # R
161 # R10, 1000
162 ADDI $t0, $zero, 267 # R
163 # R10, 1000
164 ADDI $t0, $zero, 268 # R
165 # R10, 1000
166 ADDI $t0, $zero, 269 # R
167 # R10, 1000
168 ADDI $t0, $zero, 270 # R
169 # R10, 1000
170 ADDI $t0, $zero, 271 # R
171 # R10, 1000
172 ADDI $t0, $zero, 272 # R
173 # R10, 1000
174 ADDI $t0, $zero, 273 # R
175 # R10, 1000
176 ADDI $t0, $zero, 274 # R
177 # R10, 1000
178 ADDI $t0, $zero, 275 # R
179 # R10, 1000
180 ADDI $t0, $zero, 276 # R
181 # R10, 1000
182 ADDI $t0, $zero, 277 # R
183 # R10, 1000
184 ADDI $t0, $zero, 278 # R
185 # R10, 1000
186 ADDI $t0, $zero, 279 # R
187 # R10, 1000
188 ADDI $t0, $zero, 280 # R
189 # R10, 1000
190 ADDI $t0, $zero, 281 # R
191 # R10, 1000
192 ADDI $t0, $zero, 282 # R
193 # R10, 1000
194 ADDI $t0, $zero, 283 # R
195 # R10, 1000
196 ADDI $t0, $zero, 284 # R
197 # R10, 1000
198 ADDI $t0, $zero, 285 # R
199 # R10, 1000
200 ADDI $t0, $zero, 286 # R
201 # R10, 1000
202 ADDI $t0, $zero, 287 # R
203 # R10, 1000
204 ADDI $t0, $zero, 288 # R
205 # R10, 1000
206 ADDI $t0, $zero, 289 # R
207 # R10, 1000
208 ADDI $t0, $zero, 290 # R
209 # R10, 1000
210 ADDI $t0, $zero, 291 # R
211 # R10, 1000
212 ADDI $t0, $zero, 292 # R
213 # R10, 1000
214 ADDI $t0, $zero, 293 # R
215 # R10, 1000
216 ADDI $t0, $zero, 294 # R
217 # R10, 1000
218 ADDI $t0, $zero, 295 # R
219 # R10, 1000
220 ADDI $t0, $zero, 296 # R
221 # R10, 1000
222 ADDI $t0, $zero, 297 # R
223 # R10, 1000
224 ADDI $t0, $zero, 298 # R
225 # R10, 1000
226 ADDI $t0, $zero, 299 # R
227 # R10, 1000
228 ADDI $t0, $zero, 300 # R
229 # R10, 1000
230 ADDI $t0, $zero, 301 # R
231 # R10, 1000
232 ADDI $t0, $zero, 302 # R
233 # R10, 1000
234 ADDI $t0, $zero, 303 # R
235 # R10, 1000
236 ADDI $t0, $zero, 304 # R
237 # R10, 1000
238 ADDI $t0, $zero, 305 # R
239 # R10, 1000
240 ADDI $t0, $zero, 306 # R
241 # R10, 1000
242 ADDI $t0, $zero, 307 # R
243 # R10, 1000
244 ADDI $t0, $zero, 308 # R
245 # R10, 1000
246 ADDI $t0, $zero, 309 # R
247 # R10, 1000
248 ADDI $t0, $zero, 310 # R
249 # R10, 1000
250 ADDI $t0, $zero, 311 # R
251 # R10, 1000
252 ADDI $t0, $zero, 312 # R
253
```

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:
add \$s1, \$s2, \$s3 (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.
addi \$s1, \$s2, 100 (Basic form: OP rd, rs, imm)
- **J Instructions:** instructions that jump to another memory location.
j done

Recap: MIPS Commands



The screenshot shows a MIPS simulator window. On the left, there's a text area with assembly code. On the right, there's a register window titled 'Registers' showing the state of various registers.

```
# Choose 'Hello world' as the top of the stack
addi $s0, $zero, 12 # R
$0 $0, 12
addi $s1, $zero, 181 # R
$1 $1, 181
addi $s2, $zero, 100 # 2
$2 $2, 100
addi $s3, $zero, 100 # 1
$3 $3, 100
addi $s4, $zero, 111 # R
$4 $4, 111
addi $s5, $zero, 110 # R
$5 $5, 110
addi $s6, $zero, 110 # R
$6 $6, 110
addi $s7, $zero, 111 # R
$7 $7, 111
addi $s8, $zero, 114 # R
$8 $8, 114
addi $s9, $zero, 100 # 1
$9 $9, 100
addi $s10, $zero, 100 # R
$10 $10, 100
addi $s11, $zero, 10 # 1
$11 $11, 10
addi $s12, $zero, 0 # (null)
$12 $12, 0
addi $s13, $zero, 4 # 4 is for print ending
$13 $13, 4
syscall # print to the log
```

The register window on the right shows the following values:

Register	Value
\$0	12
\$1	181
\$2	100
\$3	100
\$4	111
\$5	110
\$6	110
\$7	111
\$8	114
\$9	100
\$10	100
\$11	10
\$12	0
\$13	4

- **Registers:** locations for storing information that can be quickly accessed. Names start with '\$': \$s0, \$s1, \$t0, \$t1,...
- **R Instructions:** Commands that use data in the registers:
add \$s1, \$s2, \$s3 (Basic form: OP rd, rs, rt)
- **I Instructions:** instructions that also use intermediate values.
addi \$s1, \$s2, 100 (Basic form: OP rd, rs, imm)
- **J Instructions:** instructions that jump to another memory location.
j done (Basic form: OP label)

Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.



The screenshot shows a debugger window with two panes. The left pane displays assembly instructions, including `movl $0, %eax`, `leal 1(%eax), %eax`, and `cmpl $10, %eax`. The right pane shows the state of registers, with `%eax` containing the value 1.

Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.



Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.



Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:



Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
 - ▶ **Unconditional:** `j Done` will jump to the address with label `Done`.



Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
 - ▶ **Unconditional:** `j Done` will jump to the address with label `Done`.
 - ▶ **Branch if Equal:** `beq $s0 $s1 DoAgain` will jump to the address with label `DoAgain` if the registers `$s0` and `$s1` contain the same value.



Loops & Jumps in Machine Language

- Instead of built-in looping structures like `for` and `while`, you create your own loops by “jumping” to the location in the program.
- Can indicate locations by writing **labels** at the beginning of a line.
- Then give a command to jump to that location.
- Different kinds of jumps:
 - ▶ **Unconditional:** `j Done` will jump to the address with label `Done`.
 - ▶ **Branch if Equal:** `beq $s0 $s1 DoAgain` will jump to the address with label `DoAgain` if the registers `$s0` and `$s1` contain the same value.
 - ▶ See reading for more variations.



WeMIPS

Line: 3 dis

Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stop Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

```
1 # Store "hello world!" at the top of the stack
2 ADDUI $a0, $zero, 32 # 8
3 ROR $t0, $t0, 1
4 ADDUI $t0, $zero, 101 # e
5 DD $t0, 1($t0)
6 ADDUI $t0, $zero, 108 # l
7 DD $t0, 6($t0)
8 ADDUI $t0, $zero, 108 # l
9 DD $t0, 6($t0)
10 ADDUI $t0, $zero, 111 # o
11 DD $t0, 6($t0)
12 ADDUI $t0, $zero, 32 # (space)
13 DD $t0, 6($t0)
14 ADDUI $t0, $zero, 119 # w
15 DD $t0, 6($t0)
16 ADDUI $t0, $zero, 114 # u
17 DD $t0, 7($t0)
18 ADDUI $t0, $zero, 108 # l
19 DD $t0, 6($t0)
20 ADDUI $t0, $zero, 108 # l
21 DD $t0, 6($t0)
22 ADDUI $t0, $zero, 103 # d
23 DD $t0, 10($t0)
24 ADDUI $t0, $zero, 33 # !
25 DD $t0, 11($t0)
26 ADDUI $t0, $zero, 0 # (null)
27 DD $t0, 12($t0)
28
29 ADDUI $v0, $zero, 6 # 4 in for print string
30 ADDUI $a0, $a0, 0
31 syscall # print to the log
```

Step Run ☐ Enable auto-switching

S	T	A	V	Stack	Log
				a0:	10
				t0:	9
				a0:	9
				a0:	22
				a0:	695
				a0:	970
				a0:	927
				a0:	418

(Demo with WeMIPS)

Today's Topics



- Data Representation
- Machine Language: Jumps & Loops
- **Recap of Python & Circuits**
- Design Patterns: Sorting

Python & Circuits Review: 10 Weeks in 10 Minutes



A whirlwind tour of the semester, so far...

Week 1: print(), loops, comments, & turtles

Week 1: print(), loops, comments, & turtles

- Introduced comments & print():

```
#Name:  Thomas Hunter
```

← *These lines are comments*

```
#Date:  September 1, 2017
```

← *(for us, not computer to read)*

```
#This program prints:  Hello, World!
```

← *(this one also)*

```
print("Hello, World!")
```

← *Prints the string "Hello, World!" to the screen*

Week 1: print(), loops, comments, & turtles

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

```
#Date: September 1, 2017
```

← (for us, not computer to read)

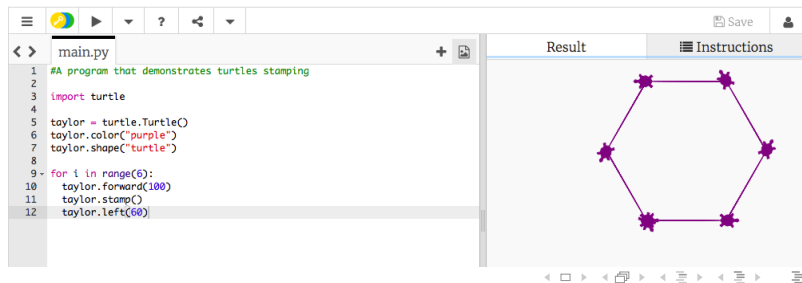
```
#This program prints: Hello, World!
```

← (this one also)

```
print("Hello, World!")
```

← Prints the string "Hello, World!" to the screen

- As well as definite loops & the turtle package:



Week 2: variables, data types, more on loops & range()

Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.

Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers

Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers

Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters

Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items

Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. `[3, 1, 4, 5, 9]` or `['violet', 'purple', 'indigo']`

Week 2: variables, data types, more on loops & range()






- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. [3, 1, 4, 5, 9] or ['violet', 'purple', 'indigo']
 - ▶ **class variables**: for complex objects, like turtles.

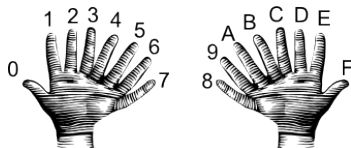
Week 2: variables, data types, more on loops & range()

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. [3, 1, 4, 5, 9] or ['violet', 'purple', 'indigo']
 - ▶ **class variables**: for complex objects, like turtles.
- More on loops & ranges:

```
1 #Predict what will be printed:
2
3 for num in [2,4,6,8,10]:
4     print(num)
5
6 sum = 0
7 for x in range(0,12,2):
8     print(x)
9     sum = sum + x
10
11 print(x)
12
13 for c in "ABCD":
14     print(c)
```

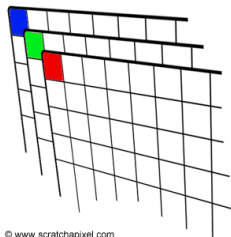
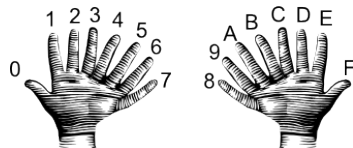
Week 3: colors, hex, slices, numpy & images

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



Week 3: colors, hex, slices, numpy & images

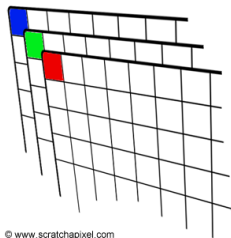
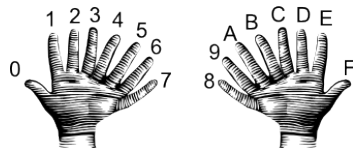
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



© www.scratchapixel.com

Week 3: colors, hex, slices, numpy & images

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



© www.scratchapixel.com

```
>>> a[0,3:5]  
array([3,4])
```

```
>>> a[4:,4:]  
array([[44, 45],  
       [54, 55]])
```

```
>>> a[:,2]  
array([2,12,22,32,42,52])
```

```
>>> a[2::2,::2]  
array([[20,22,24],  
       [40,42,44]])
```

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

Week 4: design problem (cropping images) & decisions



Week 4: design problem (cropping images) & decisions



- First: specify inputs/outputs. *Input file name, output file name, upper, lower, left, right ("bounding box")*

Week 4: design problem (cropping images) & decisions



- First: specify inputs/outputs. *Input file name, output file name, upper, lower, left, right ("bounding box")*
- Next: write pseudocode.
 - ① Import numpy and pyplot.
 - ② Ask user for file names and dimensions for cropping.
 - ③ Save input file to an array.
 - ④ Copy the cropped portion to a new array.
 - ⑤ Save the new array to the output file.

Week 4: design problem (cropping images) & decisions



- First: specify inputs/outputs. *Input file name, output file name, upper, lower, left, right ("bounding box")*
- Next: write pseudocode.
 - ① Import numpy and pyplot.
 - ② Ask user for file names and dimensions for cropping.
 - ③ Save input file to an array.
 - ④ Copy the cropped portion to a new array.
 - ⑤ Save the new array to the output file.
- Next: translate to Python.

Week 4: design problem (cropping images) & decisions

```
yearBorn = int(input('Enter year born: '))
if yearBorn < 1946:
    print("Greatest Generation")
elif yearBorn <= 1964:
    print("Baby Boomer")
elif yearBorn <= 1984:
    print("Generation X")
elif yearBorn <= 2004:
    print("Millennial")
else:
    print("TBD")

x = int(input('Enter number: '))
if x % 2 == 0:
    print('Even number')
else:
    print('Odd number')
```

Week 5: logical operators, truth tables & logical circuits

```
origin = "Indian Ocean"
winds = 100
if (winds > 74):
    print("Major storm, called a ", end="")
    if origin == "Indian Ocean" or origin == "South Pacific":
        print("cyclone.")
    elif origin == "North Pacific":
        print("typhoon.")
    else:
        print("hurricane.")

visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

Week 5: logical operators, truth tables & logical circuits

```
origin = "Indian Ocean"
winds = 100
if (winds > 74):
    print("Major storm, called a ", end="")
    if origin == "Indian Ocean" or origin == "South Pacific":
        print("cyclone.")
    elif origin == "North Pacific":
        print("typhoon.")
    else:
        print("hurricane.")

visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

in1		in2	returns:
False	and	False	False
False	and	True	False
True	and	False	False
True	and	True	True



Week 6: structured data, pandas, & more design

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,
All population figures are consistent with present-day boundaries,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,
,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island>Total
1698,4937,2017,,,727,7681
1771,21883,3623,,,2847,28423
1790,33131,45049,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,40203,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419801,45468,37393,33829,1470103
1880,1164673,599495,56559,51980,38991,1911690
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,2437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018256,469042,732016,116511,5620048
1930,1867312,2560461,1079129,1265258,159346,4590446
1940,1889924,2698295,1297634,1394711,174441,7454995
1950,1940101,2738075,1500849,1452177,191555,78931957
1960,1698281,2627319,1809578,1624815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7094862
1980,1428285,2230936,1801325,1168872,352121,7071639
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1494873,2504790,2230722,1385108,448730,8175123
2015,1644518,2636735,2339150,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,
All population figures are consistent with present-day boundaries,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,
,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island>Total
1698,4937,2017,,,727,7681
1771,21883,3623,,,2847,28423
1790,,30131,45049,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,40203,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419921,45468,37393,33829,1470193
1880,1164673,599495,56559,51980,38991,1911690
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,2437202
1910,2331542,1634351,284041,430980,85969,4766883
1920,2284103,2018296,469042,732016,116511,5620048
1930,1867312,2560461,1079129,1265258,159346,6506446
1940,1889924,2698295,1297634,1394711,174441,7454995
1950,1940101,2738275,1550849,1452177,191555,78991957
1960,1698281,2627319,1809578,1624815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1891325,1168972,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1494873,2504790,2230722,1385108,448730,8175123
2015,1644518,2636735,2339150,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
pop = pd.read_csv('nycHistPop.csv',skiprows=5)
```

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,
All population figures are consistent with present-day boundaries,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,
,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island>Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,45449,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,40203,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419921,45448,37393,33829,1470183
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,2437202
1910,2331542,1634351,284041,430980,85949,4766883
1920,2284103,2018256,469042,732016,116531,4620048
1930,1867312,2560461,1079129,1265258,158346,4590446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1940101,2738075,1550849,1451277,191555,78991957
1960,1698281,2627319,1809578,1424815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2210936,1801325,1168872,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1484873,2504760,2230722,1385108,448730,81751123
2015,1644518,2636735,2339150,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
pop = pd.read_csv('nycHistPop.csv', skiprows=5)
```

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,
All population figures are consistent with present-day boundaries,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,
,,,,,
```

```
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island>Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,45049,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,40203,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,3344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419801,45468,37393,33829,1470103
1880,1164673,599495,56559,51980,38991,1911690
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,24372702
1910,2331542,1634351,284041,430980,85949,4766883
1920,2284103,2018256,469042,732016,116511,8420048
1930,1867312,2580461,1079129,1265258,158746,6506446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1940101,2738275,1500849,1451277,291555,78991957
1960,1698281,2627319,1809578,1624815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7894862
1980,1428285,2230936,1801325,1168872,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1332650,443728,8008278
2010,1484873,2504760,2230722,1385108,448730,81751123
2015,1644518,2636735,2339150,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6

```
pop.plot(x="Year")
plt.show()
```

Week 6: structured data, pandas, & more design

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
pop = pd.read_csv('nycHistPop.csv', skiprows=5)
```

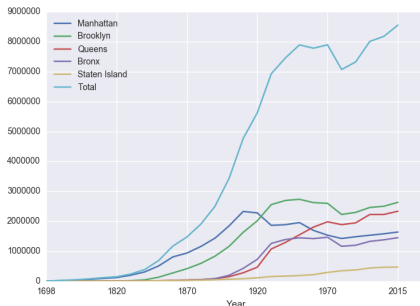
```
pop.plot(x="Year")
plt.show()
```

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,
All population figures are consistent with present-day boundaries.
First census after the consolidation of the five boroughs.

```
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,727,7681
1771,21863,3623,,2847,28423
1790,33131,4548,6159,1781,3827,49447
1800,40515,5740,6642,1755,4563,79215
1810,96373,9303,7444,2267,5347,119734
1820,123706,11187,8246,2782,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312710,47613,14480,5344,10965,391114
1850,515547,138882,18593,8032,15061,696115
1860,813649,279122,32963,23593,25492,1174779
1870,942292,419801,45468,37393,33829,1470183
1880,1164673,599495,56559,51980,38991,1911698
1890,1441216,838547,87050,88908,51692,2507414
1900,1650093,1146582,152899,200507,67021,3437202
1910,2331542,1634351,284041,430989,85969,4766883
1920,2284103,2018256,469042,732016,116531,5620048
1930,1867312,2560451,1079129,1265598,159346,4906446
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1940101,2738275,1500469,1452177,291559,7892957
1960,1698281,2627319,1809578,1624815,221993,7781984
1970,1539233,2602012,1986473,1471701,295443,7094862
1980,1428285,2230936,1801325,1164872,352121,7071439
1990,1487536,2300644,1951598,1203789,378977,7322564
2000,1537195,2465326,2229379,1326550,443728,8008278
2010,1484873,2504760,2230722,1385108,468730,8175123
2015,1644518,2636735,2339155,1455444,476558,8550405
```

nycHistPop.csv

In Lab 6



Week 7: functions

- Functions are a way to break code into pieces, that can be easily reused.

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

Week 7: functions

```
#Name:  your name here  
#Date:  October 2017  
#This program, uses functions,  
#      says hello to the world!
```

```
def main():  
    print("Hello, World!")  
  
if __name__ == "__main__":  
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.

Week 7: functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#      says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`

Week 7: functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:

Week 7: functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
Example: `print("Hello", "World")`

Week 7: functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#      says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

Week 7: functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

Week 8: function parameters, github

- Functions can have **input parameters**.

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: ' ))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: ' ))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

Week 8: function parameters, github

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: ' ))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: ' ))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

Week 8: function parameters, github

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: ' ))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: ' ))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

Week 8: function parameters, github

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: '))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: '))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**

Week 8: function parameters, github

```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: '))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: '))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**
- Functions can also **return values** to where it was called.

Week 8: function parameters, github

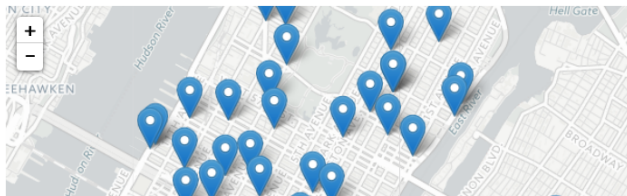
```
def totalWithTax(food,tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)  
  
lunch = float(input('Enter lunch total: '))  
lTip = float(input('Enter lunch tip: '))  
lTotal = totalWithTax(lunch, lTip)  
print('Lunch total is', lTotal)  
  
dinner= float(input('Enter dinner total: '))  
dTip = float(input('Enter dinner tip: '))  
dTotal = totalWithTax(dinner, dTip)  
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**.
- Functions can also **return values** to where it was called.

Week 9: top-down design, folium, loops, and random()



```
def main():
    dataF = getData()
    latColName, lonColName = getColumnNames()
    lat, lon = getLocale()
    cityMap = folium.Map(location = [lat,lon], tiles = 'cartodbpositron', zoom_start=11)
    dotAllPoints(cityMap,dataF,latColName,lonColName)
    markAndFindClosest(cityMap,dataF,latColName,lonColName,lat,lon)
    writeMap(cityMap)
```


Week 10: more on loops, searching data, random(), machine language

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.

```
import turtle
import random

trex = turtle.Turtle()
trex.speed(10)

for i in range(100):
    trex.forward(10)
    a = random.randrange(0,360,90)
    trex.right(a)
```

Week 10: more on loops, searching data, random(), machine language

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.
- Very useful for checking user input for correctness.

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
    trey.forward(10)
    a = random.randrange(0,360,90)
    trey.right(a)
```

Week 10: more on loops, searching data, random(), machine language

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

```
import turtle
import random

trex = turtle.Turtle()
trex.speed(10)

for i in range(100):
    trex.forward(10)
    a = random.randrange(0,360,90)
    trex.right(a)
```

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.
- Very useful for checking user input for correctness.
- Python's built-in random package has useful methods for generating random whole numbers and real numbers.

Week 10: more on loops, searching data, random(), machine language

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
    trey.forward(10)
    a = random.randrange(0,360,90)
    trey.right(a)
```

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.
- Very useful for checking user input for correctness.
- Python's built-in random package has useful methods for generating random whole numbers and real numbers.
- To use, must include:
import random.

Python & Circuits Review: 10 Weeks in 10 Minutes



- Input/Output (I/O): `input()` and `print()`;
pandas for CSV files
- Types:
 - ▶ Primitive: `int`, `float`, `bool`, `string`;
 - ▶ Container: lists (but not dictionaries/hashtes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`
- Logical Expressions & Circuits
- Functions: parameters & returns
- Packages:
 - ▶ Built-in: `turtle`, `math`, `random`
 - ▶ Popular: `numpy`, `matplotlib`, `pandas`, `folium`
- Simplified Machine Language

Today's Topics



- Data Representation
- Machine Language: Jumps & Loops
- Recap of Python & Circuits
- **Design Patterns: Sorting**






Sorting Demo

- ① Let `num` be the number of items in the list.

Sorting Demo

- ① Let `num` be the number of items in the list.
- ② Repeat `num` times:

	 Insertion	 Selection	 Bubble	 Shell	 Merge
 Random					
 Nearly Sorted					
 Reversed					
 Few Unique					

Sorting Demo

- ① Let `num` be the number of items in the list.
- ② Repeat `num` times:
 - If the first in line is taller than the second, switch places.

	 Insertion	 Selection	 Bubble	 Shell	 Merge
 Random					
 Nearly Sorted					
 Reversed					
 Few Unique					

Sorting Demo

- ① Let `num` be the number of items in the list.
- ② Repeat `num` times:
 - ▶ If the first in line is taller than the second, switch places.
 - ▶ Next check if the current second in line is taller than the third.

Sorting Demo

						
Random						
Nearly Sorted						
Reversed						
Few Unique						

① Let `num` be the number of items in the list.

② Repeat `num` times:

- ▶ If the first in line is taller than the second, switch places.
- ▶ Next check if the current second in line is taller than the third.
- ▶ If so, switch places.

Sorting Demo

- ① Let `num` be the number of items in the list.
- ② Repeat `num` times:
 - ▶ If the first in line is taller than the second, switch places.
 - ▶ Next check if the current second in line is taller than the third.
 - ▶ If so, switch places.
 - ▶ Repeat until you reach the end of the list.

Sorting Demo

	Insertion	Selection	Bubble	Shell	Merge
Random					
Nearly Sorted					
Reversed					
Few Unique					

Show sorting demo.

Lecture Slip: Design Patterns

	 Insertion	 Selection	 Bubble	 Shell	 Merge
 Random					
 Nearly Sorted					
 Reversed					
 Few Unique					

In pairs or triples:

- Fill in the UTAs' name at the top of the sheet.
- What does the code do?

Recap: Python, Languages, & Design

- On lecture slip, write down a topic you wish we had spent more time (and why).

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

Recap: Python, Languages, & Design

```
#Name: your name here  
#Date: October 2017  
#This program, uses functions,  
#    says hello to the world!
```

```
def main():  
    print("Hello, World!")
```

```
if __name__ == "__main__":  
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language

Recap: Python, Languages, & Design

```
#Name: your name here  
#Date: October 2017  
#This program, uses functions,  
#    says hello to the world!
```

```
def main():  
    print("Hello, World!")
```

```
if __name__ == "__main__":  
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language
- Logical Circuits

Recap: Python, Languages, & Design

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language
- Logical Circuits
- Simplified Machine Language

Recap: Python, Languages, & Design

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language
- Logical Circuits
- Simplified Machine Language
- Design: from written description ('specs') to function inputs & outputs ('APIs')

Recap: Python, Languages, & Design

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Python language
- Logical Circuits
- Simplified Machine Language
- Design: from written description ('specs') to function inputs & outputs ('APIs')
- Pass your lecture slips to the aisles for the UTAs to collect.

Final Overview: Top-Down Design & APIs

For each question, write **only the function header (name & inputs) and return values** (often called the Application Programming Interface (API)):

Final Overview: Top-Down Design & APIs

For each question, write **only the function header (name & inputs) and return values** (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.
- Write a function that takes a string and returns its length.
- Write a function that, given a DataFrame, returns the minimal value in the first column.
- Write a function that takes a whole number and returns the corresponding binary number as a string.
- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

Final Overview: Top-Down Design & APIs

For each question, write **only the function header (name & inputs) and return values** (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.
- Write a function that takes a string and returns its length.
- Write a function that, given a DataFrame, returns the minimal value in the first column.
- Write a function that takes a whole number and returns the corresponding binary number as a string.
- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

(Hint: highlight key words, make list of inputs, list of outputs, then put together.)

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

```
def kg2lbs(kg):  
    ...  
    return(lbs)
```

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

```
def kg2lbs(kg)
    lbs = kg * 2.2
    return(lbs)
```

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

```
def sLength(str):  
    ...  
    return(length)
```

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

```
def sLength(str):  
    length = len(str)  
    return(length)
```

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

```
def getMin(df):  
    ...  
    return(min)
```

Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

```
def getMin(df):  
    min = df['Manhattan'].min()  
    return(min)
```


Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

```
def num2bin(num):  
    ...  
    return(bin)
```

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

```
def num2bin(num):  
    binStr = ""  
    while (num > 0):  
        #Divide by 2, and add the remainder to the string  
        r = num %2  
        binString = str(r) + binStr  
        num = num / 2  
    return(binStr)
```

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

Final Overview

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

```
def computePayment(loan,rate,year):  
    ....  
    return(payment)
```

Final Overview

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

```
def computePayment(loan,rate,year):  
    (Some formula for payment)  
    return(payment)
```