

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

- This lecture will be recorded

Announcements

- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**



Announcements

- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?



Announcements



- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?
 - ▶ Get an A (+)

Announcements



- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?
 - ▶ Get an A (+)
 - ▶ Be very comfortable with our labs and programming assignments

Announcements



- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?
 - ▶ Get an A (+)
 - ▶ Be very comfortable with our labs and programming assignments
 - ▶ Be able to describe a successful tutoring experience (go to tutoring!!!)

Announcements



- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?
 - ▶ Get an A (+)
 - ▶ Be very comfortable with our labs and programming assignments
 - ▶ Be able to describe a successful tutoring experience (go to tutoring!!!)
 - ▶ Previous tutoring experience helpful but not expected

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, 20 December, 9-11am.

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, 20 December, 9-11am.

The early final exam (alternative date) is on Friday, 17 December, 8-10am.

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, 20 December, 9-11am.

The early final exam (alternative date) is on Friday, 17 December, 8-10am.

Instead of a review sheet, we have:

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, 20 December, 9-11am.

The early final exam (alternative date) is on Friday, 17 December, 8-10am.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, 20 December, 9-11am.

The early final exam (alternative date) is on Friday, 17 December, 8-10am.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *UTAs in drop-in tutoring happy to review concepts and old exam questions.*

Frequently Asked Questions

From email and tutoring.

- **When is the final? Is there a review sheet?**

The official final is Monday, 20 December, 9-11am.

The early final exam (alternative date) is on Friday, 17 December, 8-10am.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *UTAs in drop-in tutoring happy to review concepts and old exam questions.*
- ▶ *There will be opportunity for **in-person practice** during our last meeting on 7 December.*

Today's Topics



- Design Patterns: Searching
- Python Recap
- Machine Language
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

Today's Topics



- **Design Patterns: Searching**
 - Python Recap
 - Machine Language
 - Machine Language: Jumps & Loops
 - Binary & Hex Arithmetic
 - Final Exam: Format

Predict what the code will do:

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')|
```

Python Tutor

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

(Demo with pythonTutor)

Design Pattern: Linear Search

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

- Example of **linear search**.

Design Pattern: Linear Search

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

- Example of **linear search**.
- Start at the beginning of the list.

Design Pattern: Linear Search

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

- Example of **linear search**.
- Start at the beginning of the list.
- Look at each item, one-by-one.

Design Pattern: Linear Search

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

- Example of **linear search**.
- Start at the beginning of the list.
- Look at each item, one-by-one.
- Stop when found, or the end of list is reached.

Today's Topics



- Design Patterns: Searching
- **Python Recap**
- Machine Language
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic

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:

The screenshot shows a code editor interface with a toolbar at the top. The file tab shows 'main.py'. The code area contains the following Python script:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

To the right of the code editor is a results panel titled 'Result' which displays a purple hexagon drawn by the turtle. Each vertex of the hexagon has a small purple star-like stamp.

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.

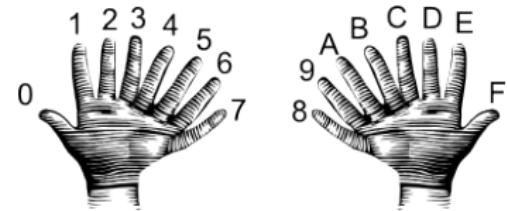
Week 2: variables, data types, more on loops & range()

- 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(sum)  
12  
13 for c in "ABCD":  
14     print(c)
```

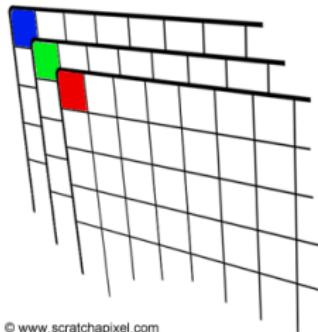
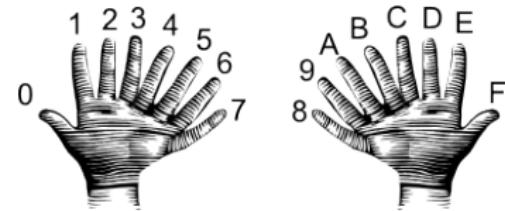
Week 3: colors, hex, slices, numpy & images

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



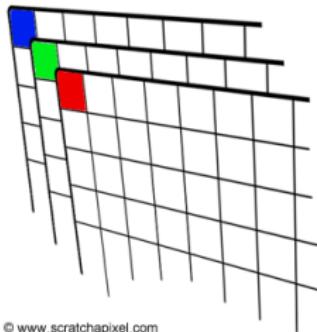
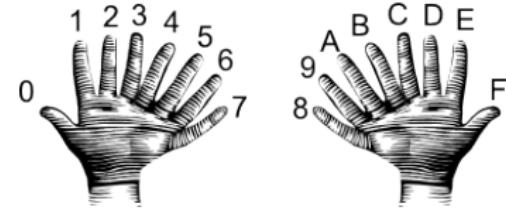
Week 3: colors, hex, slices, numpy & images

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



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

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



```
>>> 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.")

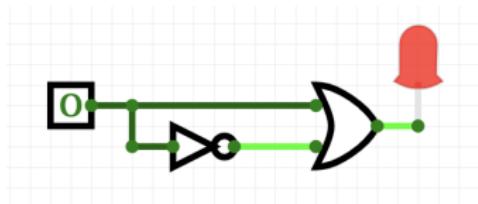
visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

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	and	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
1890,1,093,703,727,718,2,538,2,037
1891,1,093,703,727,718,2,538,2,037
1892,1,093,703,727,718,2,538,2,037
1893,1,093,703,727,718,2,538,2,037
1894,1,093,703,727,718,2,538,2,037
1895,1,093,703,727,718,2,538,2,037
1896,1,093,703,727,718,2,538,2,037
1897,1,093,703,727,718,2,538,2,037
1898,1,093,703,727,718,2,538,2,037
1899,1,093,703,727,718,2,538,2,037
1900,1,093,703,727,718,2,538,2,037
1901,1,093,703,727,718,2,538,2,037
1902,1,093,703,727,718,2,538,2,037
1903,1,093,703,727,718,2,538,2,037
1904,1,093,703,727,718,2,538,2,037
1905,1,093,703,727,718,2,538,2,037
1906,1,093,703,727,718,2,538,2,037
1907,1,093,703,727,718,2,538,2,037
1908,1,093,703,727,718,2,538,2,037
1909,1,093,703,727,718,2,538,2,037
1910,1,093,703,727,718,2,538,2,037
1911,1,093,703,727,718,2,538,2,037
1912,1,093,703,727,718,2,538,2,037
1913,1,093,703,727,718,2,538,2,037
1914,1,093,703,727,718,2,538,2,037
1915,1,093,703,727,718,2,538,2,037
1916,1,093,703,727,718,2,538,2,037
1917,1,093,703,727,718,2,538,2,037
1918,1,093,703,727,718,2,538,2,037
1919,1,093,703,727,718,2,538,2,037
1920,1,093,703,727,718,2,538,2,037
1921,1,093,703,727,718,2,538,2,037
1922,1,093,703,727,718,2,538,2,037
1923,1,093,703,727,718,2,538,2,037
1924,1,093,703,727,718,2,538,2,037
1925,1,093,703,727,718,2,538,2,037
1926,1,093,703,727,718,2,538,2,037
1927,1,093,703,727,718,2,538,2,037
1928,1,093,703,727,718,2,538,2,037
1929,1,093,703,727,718,2,538,2,037
1930,1,093,703,727,718,2,538,2,037
1931,1,093,703,727,718,2,538,2,037
1932,1,093,703,727,718,2,538,2,037
1933,1,093,703,727,718,2,538,2,037
1934,1,093,703,727,718,2,538,2,037
1935,1,093,703,727,718,2,538,2,037
1936,1,093,703,727,718,2,538,2,037
1937,1,093,703,727,718,2,538,2,037
1938,1,093,703,727,718,2,538,2,037
1939,1,093,703,727,718,2,538,2,037
1940,1,093,703,727,718,2,538,2,037
1941,1,093,703,727,718,2,538,2,037
1942,1,093,703,727,718,2,538,2,037
1943,1,093,703,727,718,2,538,2,037
1944,1,093,703,727,718,2,538,2,037
1945,1,093,703,727,718,2,538,2,037
1946,1,093,703,727,718,2,538,2,037
1947,1,093,703,727,718,2,538,2,037
1948,1,093,703,727,718,2,538,2,037
1949,1,093,703,727,718,2,538,2,037
1950,1,093,703,727,718,2,538,2,037
1951,1,093,703,727,718,2,538,2,037
1952,1,093,703,727,718,2,538,2,037
1953,1,093,703,727,718,2,538,2,037
1954,1,093,703,727,718,2,538,2,037
1955,1,093,703,727,718,2,538,2,037
1956,1,093,703,727,718,2,538,2,037
1957,1,093,703,727,718,2,538,2,037
1958,1,093,703,727,718,2,538,2,037
1959,1,093,703,727,718,2,538,2,037
1960,1,093,703,727,718,2,538,2,037
1961,1,093,703,727,718,2,538,2,037
1962,1,093,703,727,718,2,538,2,037
1963,1,093,703,727,718,2,538,2,037
1964,1,093,703,727,718,2,538,2,037
1965,1,093,703,727,718,2,538,2,037
1966,1,093,703,727,718,2,538,2,037
1967,1,093,703,727,718,2,538,2,037
1968,1,093,703,727,718,2,538,2,037
1969,1,093,703,727,718,2,538,2,037
1970,1,093,703,727,718,2,538,2,037
1971,1,093,703,727,718,2,538,2,037
1972,1,093,703,727,718,2,538,2,037
1973,1,093,703,727,718,2,538,2,037
1974,1,093,703,727,718,2,538,2,037
1975,1,093,703,727,718,2,538,2,037
1976,1,093,703,727,718,2,538,2,037
1977,1,093,703,727,718,2,538,2,037
1978,1,093,703,727,718,2,538,2,037
1979,1,093,703,727,718,2,538,2,037
1980,1,093,703,727,718,2,538,2,037
1981,1,093,703,727,718,2,538,2,037
1982,1,093,703,727,718,2,538,2,037
1983,1,093,703,727,718,2,538,2,037
1984,1,093,703,727,718,2,538,2,037
1985,1,093,703,727,718,2,538,2,037
1986,1,093,703,727,718,2,538,2,037
1987,1,093,703,727,718,2,538,2,037
1988,1,093,703,727,718,2,538,2,037
1989,1,093,703,727,718,2,538,2,037
1990,1,093,703,727,718,2,538,2,037
1991,1,093,703,727,718,2,538,2,037
1992,1,093,703,727,718,2,538,2,037
1993,1,093,703,727,718,2,538,2,037
1994,1,093,703,727,718,2,538,2,037
1995,1,093,703,727,718,2,538,2,037
1996,1,093,703,727,718,2,538,2,037
1997,1,093,703,727,718,2,538,2,037
1998,1,093,703,727,718,2,538,2,037
1999,1,093,703,727,718,2,538,2,037
2000,1,093,703,727,718,2,538,2,037
2001,1,093,703,727,718,2,538,2,037
2002,1,093,703,727,718,2,538,2,037
2003,1,093,703,727,718,2,538,2,037
2004,1,093,703,727,718,2,538,2,037
2005,1,093,703,727,718,2,538,2,037
2006,1,093,703,727,718,2,538,2,037
2007,1,093,703,727,718,2,538,2,037
2008,1,093,703,727,718,2,538,2,037
2009,1,093,703,727,718,2,538,2,037
2010,1,093,703,727,718,2,538,2,037
2011,1,093,703,727,718,2,538,2,037
2012,1,093,703,727,718,2,538,2,037
2013,1,093,703,727,718,2,538,2,037
2014,1,093,703,727,718,2,538,2,037
2015,1,093,703,727,718,2,538,2,037
2016,1,093,703,727,718,2,538,2,037
2017,1,093,703,727,718,2,538,2,037
2018,1,093,703,727,718,2,538,2,037
2019,1,093,703,727,718,2,538,2,037
2020,1,093,703,727,718,2,538,2,037
2021,1,093,703,727,718,2,538,2,037

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.....
Five census after the consolidation of the five boroughs.....
.....
Year,Bronx,Brooklyn,Queens,Bronx,Staten Island,Totals
1890,4937,2037,,727,7881,28423
1870,33131,4549,6159,1781,3827,49447
1860,60515,5740,6442,1755,4543,75915
1850,55349,5254,5893,1541,39734
1820,123704,11187,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,18013,14049,5348,10965,391114
1850,35549,21890,18951,5348,10965,391114
1860,613469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51980,33029,1911803
1890,1285093,616582,58748,51864,33029,1911803
1900,185093,116582,152999,200567,67621,2437202
1910,2233142,1634351,284041,430980,8569,4766803
1920,2210103,2018354,446031,446031,732013,116515,591048
1930,1667137,1796128,2079128,2079128,15831,4930446
1940,1889924,2469285,1297634,1394711,1374441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1696101,2319319,1890941,1460711,191555,7981984
1970,1539231,2460711,1471701,1471701,135443,7981984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,439787,77322564
2000,1537195,2485326,2229379,1332450,439787,8080879
2010,1583873,2504705,2272722,1385108,439787,8175133
2015,1444018,2646733,2339150,1459446,474558,8059405

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,Bronx,Brooklyn,Queens,Bronx,Staten Island,Totals  
1690,4937,2037,...,727,7881,10700  
1771,21843,36232,...,2847,28423  
1790,33131,4549,...,6159,1781,3827,49447  
1800,60515,5740,...,6442,1755,4543,75935  
1810,67000,62000,...,6442,1755,4543,75934  
1820,123704,11187,...,8246,2792,6135,152056  
1830,20589,...,20535,...,9049,...,3023,7082,242278  
1840,31100,...,10113,...,14000,...,3348,...,10965,...,391114  
1850,35549,...,128000,...,18500,...,3348,...,10965,...,391114  
1860,...,13449,...,279122,...,23903,...,23932,...,25492,...,174777  
1870,...,942292,...,419921,...,85468,...,37393,...,33029,...,1479103  
1880,...,114473,...,59943,...,56568,...,51980,...,33029,...,1911801  
1890,...,114473,...,59943,...,56568,...,51980,...,33029,...,1911801  
1900,...,185093,...,116582,...,152999,...,200567,...,67921,...,2437202  
1910,...,233142,...,1634351,...,2841,...,430980,...,8569,...,476683  
1920,...,2231103,...,2018354,...,446071,...,732013,...,11651,...,593083  
1930,...,1657110,...,1579128,...,1579128,...,1579128,...,1579128,...,4930446  
1940,...,1889924,...,2469285,...,1297634,...,1394711,...,1374441,...,7454995  
1950,...,1960101,...,2738175,...,1550949,...,1451277,...,191555,...,7991957  
1960,...,1690101,...,2738175,...,1550949,...,1451277,...,191555,...,7991957  
1970,...,1359231,...,2469285,...,1472701,...,1472701,...,135443,...,7071984  
1980,...,1426285,...,2230936,...,1891325,...,1168972,...,252121,...,7071639  
1990,...,1487536,...,2300664,...,1951598,...,1320789,...,2329777,...,7322564  
2000,...,1537195,...,2485326,...,2229379,...,1332450,...,419728,...,8080879  
2010,...,1583873,...,2504705,...,2272722,...,1385108,...,419728,...,8175133  
2015,...,1444918,...,2546733,...,2339150,...,1459444,...,474558,...,8059405
```

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,Population  
1690,203,2037,...,727,7181  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,71021,6250,6840,1803,4937,85934  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31510,19113,14049,5346,10965,391114  
1850,35549,21890,18859,5815,12581,45115  
1860,813469,279122,23903,23933,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1375000,710000,65000,58000,34000,1911814  
1900,1850993,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44601,72021,11651,50048  
1930,2667113,2079128,1578128,253454,5821,4930446  
1940,1889924,2690285,1297634,1394711,1374441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690101,2738175,1550949,1451277,191555,7981984  
1970,1539231,2465701,1472701,1235443,7981984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,737977,7322564  
2000,1537195,2485326,2223379,1332650,419728,8080879  
2010,1583873,2504705,2272722,1385108,419728,8175133  
2015,1444518,2504733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

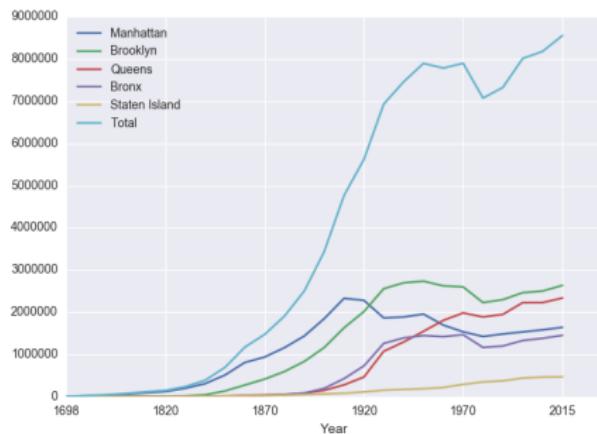
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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,Population  
1699,4937,2017,...,727,7181  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,62000,68000,1800,49734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,19113,14000,5348,10965,391114  
1850,355441,21800,18500,5850,12000,43115  
1860,613469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59940,5653,51980,33051,1911801  
1890,1370000,718000,68000,58000,35000,215134  
1900,185093,116582,152999,200567,67821,2437202  
1910,233142,1634351,2841,430980,85869,476683  
1920,2210103,2018354,44601,732013,11651,50048  
1930,2667137,2485231,479128,52543,15837,693446  
1940,188924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7891957  
1960,1696000,2319319,1809000,1460000,1800000,781984  
1970,1539231,2465071,1471701,1235443,798462  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2216722,1385108,451728,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

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

```
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).

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**.

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)
                                Actual Parameters

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 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, max design pattern, random()

```
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

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, max design pattern, random()

```
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, max design pattern, random()

```
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.

Week 10: more on loops, max design pattern, random()

```
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`.

Week 10: more on loops, max design pattern, random()

```
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`.
- The max design pattern provides a template for finding maximum value from a list.

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/hashes 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`

Lecture Quiz

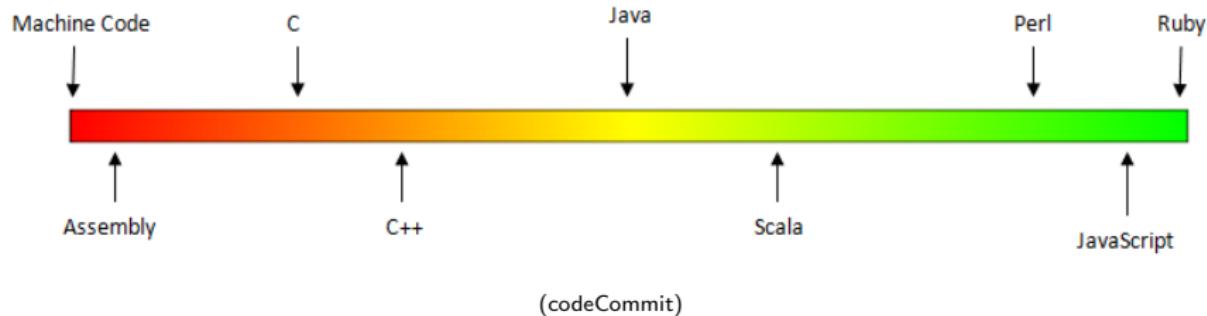
- Log-in to Gradescope
- Find LECTURE 11 Quiz
- Take the quiz
- **You have 3 minutes**

Today's Topics



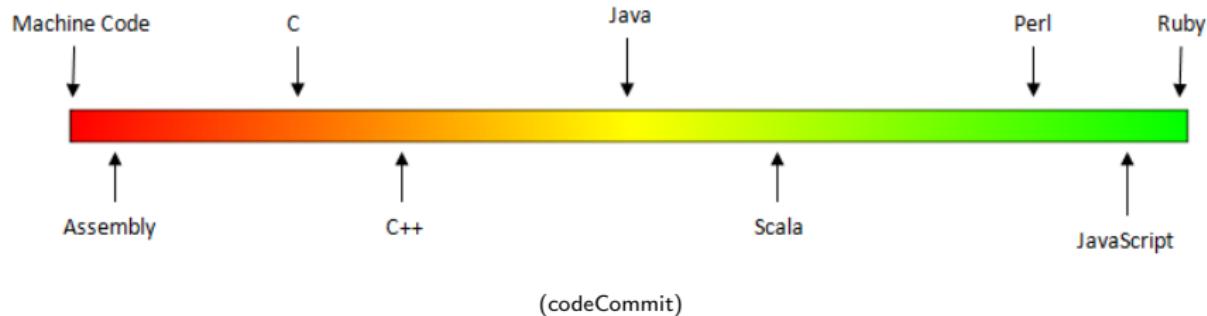
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic

Low-Level vs. High-Level Languages



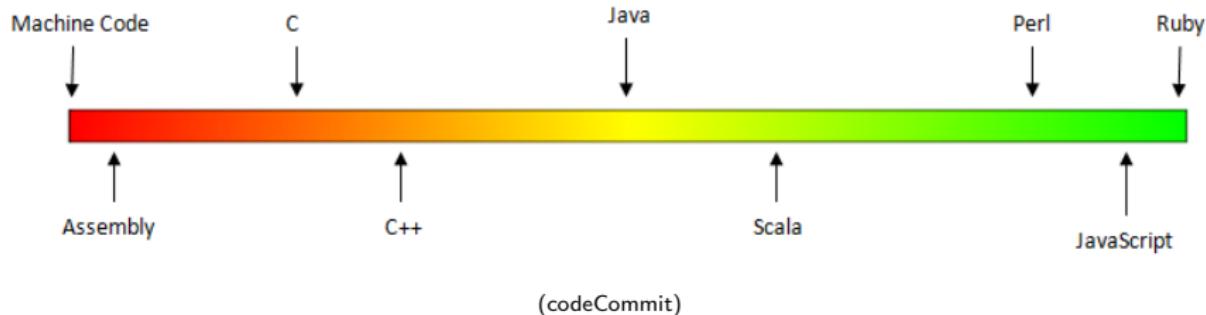
- Can view programming languages on a continuum.

Low-Level vs. High-Level Languages



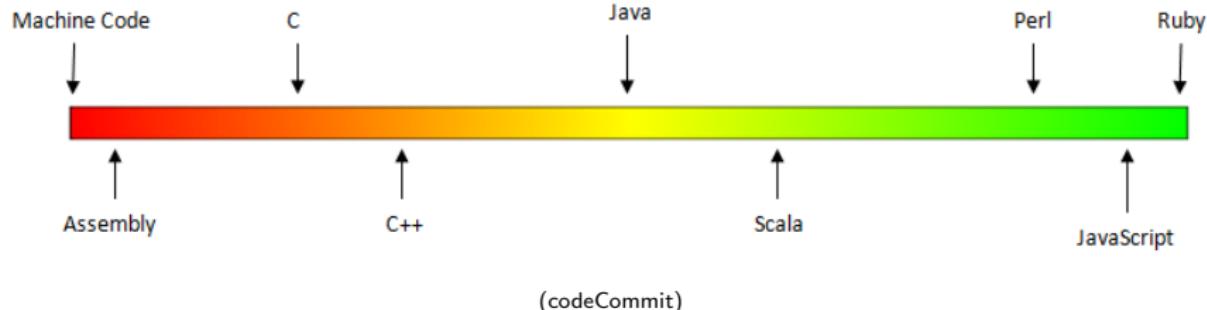
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

Low-Level vs. High-Level Languages



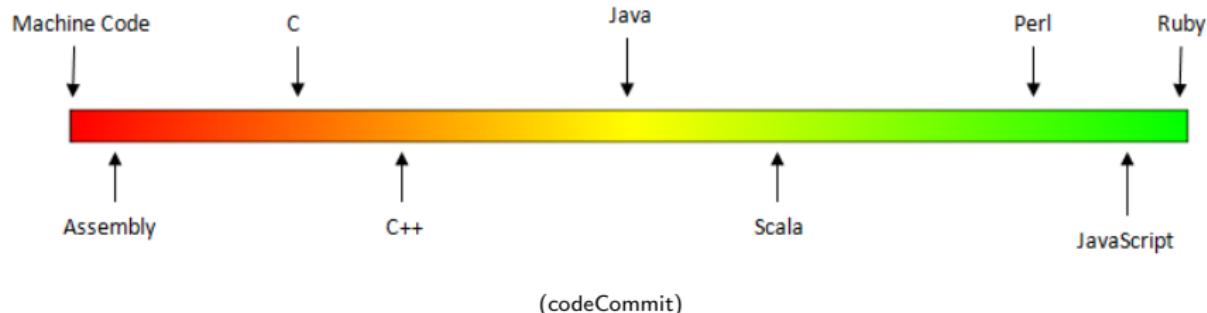
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

Low-Level vs. High-Level Languages



- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine details, such as complex variables, functions and looping that do not translate directly into machine code) are called **high-level languages**.

Low-Level vs. High-Level Languages



- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine details, such as complex variables, functions and looping that do not translate directly into machine code) are called **high-level languages**.
- Some languages, like C, are in between— allowing both low level access and high level data structures.

Processing

Bei [Lernzettel](#) für [Blindtext](#). "Blindtext" ist ein Text, der so gestaltet ist, dass er von den Augen nicht gelesen werden kann, aber von den Hörern verstanden werden kann. Der Text besteht aus einer Menge von Zeichen, die in einem speziellen Code dargestellt sind, der für die Leserichtung optimiert ist.

Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ableSEN, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt. Dies ist ein Blindtext. An ihm lässt sich

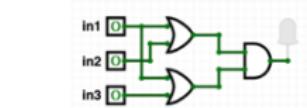


Data
&
Instructions

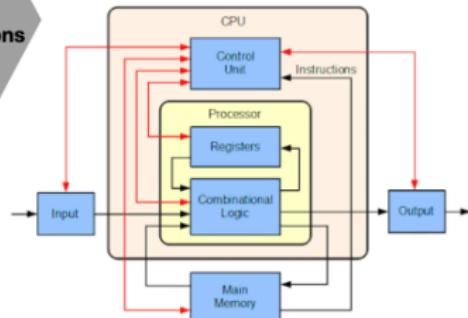
```
01110100011100100110000101  
1110010001101101110101011  
00111110001101000 101011  
00101110 00010000111111  
1110110111101000111011011  
001011010 101101 10011000  
01100101011010001000001  
11100100001101 00111101  
011000101101100011010101  
01000100000011000100 000  
0110010101 100110100101  
10010 100000110110011101  
01101100110010001101  
011000110 101110111001100  
01100110 10111011101101101  
100 01 001101100100100000  
0110011000100000110 1001  
1001011000100000110 1001  
0110110011001000110 1001  
01100110 0110110111011011  
0110000011101000101011100  
00010 110010011 100100100  
011010001011011011 01000  
1 1101101110011101001001  
0111001001011100 10001101  
00000110110001 10110 0011
```

```
def totalWithTax(food, tip):  
    total = 0  
    tax = 0.0875  
    total = food + food * tax  
    total = total + tip  
    return(total)
```

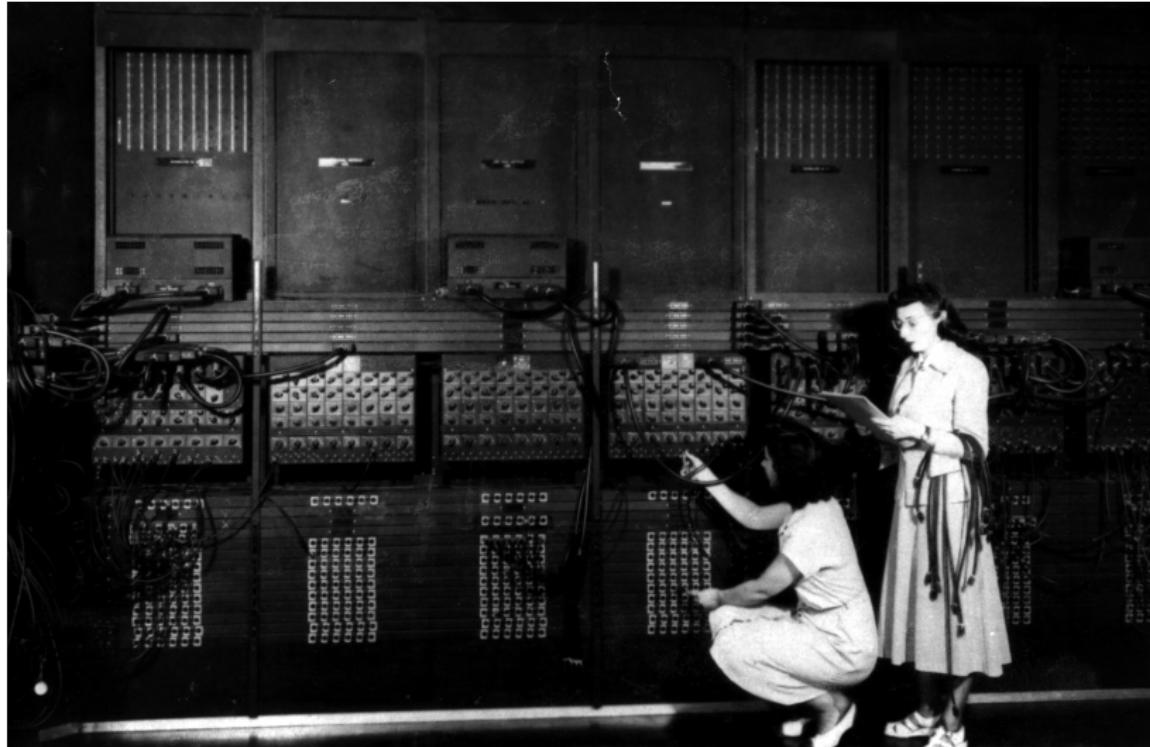
Data
&
Instructions



Circuits (switches)
On/Off 1/0 Logic
Billions of switches/bits



Machine Language



(Ruth Gordon & Ester Gerston programming the ENIAC, UPenn)

Machine Language

```
I FDX 12:01a 23- 1
A 002000 C2 30      REP #$30
A 002002 18          CLC
A 002003 F8          SED
A 002004 A9 34 12    LDA #$1234
A 002007 69 21 43    ADC #$4321
A 00200A 8F 03 7F 01 STA $017F03
A 00200E D8          CLD
A 00200F E2 30      SEP #$30
A 002011 00          BRK
A 2012

r
PB PC  NUMxDIZC .A .X .Y SP DP DB
; 00 E012 00110000 0000 0000 0002 CFFF 0000 00
g 2000

BREAK

PB PC  NUMxDIZC .A .X .Y SP DP DB
; 00 2013 00110000 5555 0000 0002 CFFF 0000 00
m 7f03 7f03
>007F03 55 55 00 00 00 00 00 00 00 00 00 00 00 00:UU .....
```

(wiki)

Machine Language

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

```

A R 002100 C2 30 REP #30
A R 002102 IB CLC
A R 002103 FB SED
A R 002104 00 24 JZ LBR #1234
A R 002105 00 21 FF #004321
A R 002106 0F 03 7F 01 STA #0017FF03
A R 002107 DD GLD
A R 002108 E2 30 SEP #38
A R 002111 00 00 00
A 2012

F
PB PC Mm=012C A X Y SP DP IB
: 00 0012 00110000 0000 0000 0002 CF7F 0000 00
6 0000

BREAK

PB PC Mm=012C A X Y SP DP IB
: 00 2013 00110000 5555 0000 0002 CF7F 0000 00
n 7103 7103
0017FF03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

(wiki)

Machine Language

- 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.



The screenshot shows a terminal window with assembly code and its corresponding binary output. The assembly code includes instructions like LDI, ADD, SUB, and MUL. The binary output consists of two columns of hex values.

PC	PC	Memory[PC]	A	X	Y	SP	DP	R
00	0012	00100000 0000 0000 0002 CFFF 0000 00						
00	0000							
BREAK								
00	0013	00100000 5555 0000 000C 1FFF 0000 00						
00	7FFF	0003 7FFF						
00	00FF	00FF 00FF						

(wiki)

Machine Language



The screenshot shows a terminal window with assembly code and its corresponding binary output. The assembly code includes instructions like LDI, GCD, LDH, ADC, STA, CLD, SED, and SWI. The binary output consists of two columns of hex values.

Assembly	Binary
LDI \$43, \$A1	00 2000 C2 3B
GCD	00 2002 1B
LDH \$C, \$D	00 2003 FB
ADC \$B, \$C	00 2004 34 12
LDH \$A, \$B	00 2005 69 21 43
ADC \$C, \$D	00 2006 40 C0 #4324
STA \$A, \$B	00 2007 8F 03 7F 01
CLD	00 2008 80 1F F0 03
SED	00 2009 E2 30
SWI	00 200A 90 M38
BREAK	00 2012

(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.

Machine Language

```

A 002300 C2 3B REP #3B
A 002302 CLC
A 002303 SED
A 002304 D2 12 LM #101234
A 002305 B2 24 42 LM #101234
A 002306 B2 7F 01 STA #0017FF
A 002307 D0 C1D CLD
A 002308 E2 3B SEP #3B
A 002311 BB 89K
A 002312

```

(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.
 - More in future architecture classes....

"Hello World!" in Simplified Machine Language

Line: 3 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav 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 ADDI $sp, $sp, -13
3 ADDI $t0, $zero, 72 # H
4 SB $t0, 0($sp)
5 ADDI $t0, $zero, 101 # e
6 SB $t0, 1($sp)
7 ADDI $t0, $zero, 108 # l
8 SB $t0, 2($sp)
9 ADDI $t0, $zero, 108 # i
10 SB $t0, 3($sp)
11 ADDI $t0, $zero, 111 # o
12 SB $t0, 4($sp)
13 ADDI $t0, $zero, 32 # (space)
14 SB $t0, 5($sp)
15 ADDI $t0, $zero, 119 # w
16 SB $t0, 6($sp)
17 ADDI $t0, $zero, 111 # o
18 SB $t0, 7($sp)
19 ADDI $t0, $zero, 114 # r
20 SB $t0, 8($sp)
21 ADDI $t0, $zero, 108 # l
22 SB $t0, 9($sp)
23 ADDI $t0, $zero, 100 # d
24 SB $t0, 10($sp)
25 ADDI $t0, $zero, 33 # !
26 SB $t0, 11($sp)
27 ADDI $t0, $zero, 0 # (null)
28 SB $t0, 12($sp)
29
30 ADDI $v0, $zero, 4 # 4 is for print string
31 ADDI $a0, $sp, 0
32 syscall           # print to the log
```

Step Run Enable auto switching

S T A V Stack Log

s0:	10
s1:	9
s2:	9
s3:	22
s4:	696
s5:	976
s6:	927
s7:	418

(WeMIPS)

WeMIPS

(Demo with WeMIPS)

MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there's a menu bar with 'File', 'Edit', 'Run', 'Help', 'Show/Hide Demo', 'Addition', 'Subtraction', 'Multiplication', 'Division', 'Hello World', 'Code Gen Base String', 'Interactive', 'Binary Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with 'Step', 'Run', 'Break', 'Create auto watching', and 'Stop' buttons. To the right of the toolbar is a status bar with 'User Guide | Unit Tests | Docs'. The main area contains assembly code and a register viewer.

```
# Shows "Hello world!" at the top of the stack
1.  li    $t0, 8192
2.  sb    $t0, 11992
3.  li    $t1, 11992
4.  sb    $t1, 11992
5.  li    $t2, 11992
6.  sb    $t2, 11992
7.  li    $t3, 11992
8.  sb    $t3, 11992
9.  li    $t4, 11992
10.  sb   $t4, 11992
11.  li    $t5, 11992
12.  sb   $t5, 11992
13.  addi $t6, $t0, tenv, 32 # (spnew)
14.  addi $t6, $t6, tenv, 128 # w
15.  addi $t6, $t6, tenv, 112 # x
16.  addi $t6, $t6, tenv, 111 # y
17.  addi $t6, $t6, tenv, 110 # z
18.  addi $t6, $t6, tenv, 113 # r
19.  addi $t6, $t6, tenv, 114 # s
20.  addi $t6, $t6, tenv, 115 # t
21.  addi $t6, $t6, tenv, 116 # u
22.  addi $t6, $t6, tenv, 117 # v
23.  addi $t6, $t6, tenv, 118 # w
24.  addi $t6, $t6, tenv, 119 # x
25.  addi $t6, $t6, tenv, 120 # y
26.  addi $t6, $t6, tenv, 121 # z
27.  addi $t6, $t6, tenv, 0 # (real)
28.  addi $t6, $t6, tenv, 122 # p
29.  addi $t6, $t6, tenv, 4 # & is for print string
30.  addi $t6, $t6, tenv, 0 # point to the log
31.  syscall
```

S	T	A	V	Stack	Log
x0	10				
x1	9				
x2	8				
x3	22				
x4	000				
x5	819				
x6	827				
x7	411				

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

MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there are tabs for "ShowHide Demo", "User Guide", "Unit Tests", and "Docs". Below the tabs are buttons for "Addition", "Subtraction", "Max", "Looper", "Stack Test", and "Hello World". There are also buttons for "Code Gen", "Save String", "Interactive", "Decimal Decimal", and "Decimal Binary". A "Debug" button is also present.

The main area displays the following assembly code:

```
1 # Ensure '$main_stack' is at the top of the stack
2
3 # SB $t0, $t1($sp), $t1 # v
4 # SB $t1, $t1($sp)
5 # SB $t0, $t1($sp), $t1 # v
6 # SB $t1, $t1($sp)
7 # SB $t0, $t1($sp), $t1 # v
8 # SB $t1, $t1($sp)
9 # SB $t0, $t1($sp), $t1 # v
10 # SB $t1, $t1($sp)
11 # ADDI $t0, $t0, 100 # (spnew)
12 # ADDI $t0, $t0, 100 # (spnew)
13 # ADDI $t0, $t0, 100 # (spnew)
14 # ADDI $t0, $t0, 100 # (spnew)
15 # ADDI $t0, $t0, 100 # (spnew)
16 # ADDI $t0, $t0, 100 # (spnew)
17 # ADDI $t0, $t0, 100 # (spnew)
18 # ADDI $t0, $t0, 100 # (spnew)
19 # ADDI $t0, $t0, 100 # (spnew)
20 # ADDI $t0, $t0, 100 # (spnew)
21 # ADDI $t0, $t0, 100 # (spnew)
22 # SB $t0, $t1($sp), $t1 # v
23 # SB $t1, $t1($sp)
24 # SB $t0, $t1($sp), $t1 # v
25 # SB $t1, $t1($sp)
26 # SB $t0, $t1($sp), $t1 # v
27 # SB $t1, $t1($sp)
28 # SB $t0, $t1($sp), $t1 # v
29 # SB $t1, $t1($sp)
30 # SB $t0, $t1($sp), $t1 # v
31 # ADDI $t0, $t0, 4 # $t1 is for print string
32 # ADDI $t0, $t0, 0 # result
33 # System call
```

Below the code, there is a register dump table with columns for S, T, V, Stack, and Log. The registers shown are \$t0 through \$t1, with their values being 10, 9, 22, 000, 879, 827, and 411 respectively.

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

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:

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

MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there's a menu bar with 'File', 'Edit', 'Get', 'Show/Hide Demo', 'User Guide | Unit Tests | Docs', and tabs for 'Assembly', 'Assembler', 'Run', 'Looper', 'Stack View', 'Hello World', 'Code Gen', 'Save String', 'Interactive', 'Decimal Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a code editor window containing the following assembly code:

```
1 # Shows "Hello world" at the top of the stack
2
3 .text
4 .globl _start
5
6 .data
7 _str: .asciz "Hello world\n"
8
9 .text
10 _start:
11     li $t0, _str
12     li $t1, 115901
13     li $t2, 115902
14     li $t3, 115903
15     li $t4, 115904
16     li $t5, 115905
17     li $t6, 115906
18     li $t7, 115907
19     li $t8, 115908
20     li $t9, 115909
21     li $t10, 115910
22     li $t11, 115911
23     li $t12, 115912
24     li $t13, 115913
25     li $t14, 115914
26     li $t15, 115915
27     li $t16, 115916
28     li $t17, 115917
29     li $t18, 115918
30     li $t19, 115919
31     li $t20, 115920
32     li $t21, 115921
33     li $t22, 115922
34     li $t23, 115923
35     li $t24, 115924
36     li $t25, 115925
37     li $t26, 115926
38     li $t27, 115927
39     li $t28, 115928
40     li $t29, 115929
41     li $t30, 115930
42
43     add $t0, $t1, $t2
44     add $t0, $t3, $t4
45     add $t0, $t5, $t6
46     add $t0, $t7, $t8
47     add $t0, $t9, $t10
48     add $t0, $t11, $t12
49     add $t0, $t13, $t14
50     add $t0, $t15, $t16
51     add $t0, $t17, $t18
52     add $t0, $t19, $t20
53     add $t0, $t21, $t22
54     add $t0, $t23, $t24
55     add $t0, $t25, $t26
56     add $t0, $t27, $t28
57     add $t0, $t29, $t30
58
59     add $t0, $t1, $t2
60     add $t0, $t3, $t4
61     add $t0, $t5, $t6
62     add $t0, $t7, $t8
63     add $t0, $t9, $t10
64     add $t0, $t11, $t12
65     add $t0, $t13, $t14
66     add $t0, $t15, $t16
67     add $t0, $t17, $t18
68     add $t0, $t19, $t20
69     add $t0, $t21, $t22
70     add $t0, $t23, $t24
71     add $t0, $t25, $t26
72     add $t0, $t27, $t28
73     add $t0, $t29, $t30
74
75     add $t0, $t1, $t2
76     add $t0, $t3, $t4
77     add $t0, $t5, $t6
78     add $t0, $t7, $t8
79     add $t0, $t9, $t10
80     add $t0, $t11, $t12
81     add $t0, $t13, $t14
82     add $t0, $t15, $t16
83     add $t0, $t17, $t18
84     add $t0, $t19, $t20
85     add $t0, $t21, $t22
86     add $t0, $t23, $t24
87     add $t0, $t25, $t26
88     add $t0, $t27, $t28
89     add $t0, $t29, $t30
90
91     add $t0, $t1, $t2
92     add $t0, $t3, $t4
93     add $t0, $t5, $t6
94     add $t0, $t7, $t8
95     add $t0, $t9, $t10
96     add $t0, $t11, $t12
97     add $t0, $t13, $t14
98     add $t0, $t15, $t16
99     add $t0, $t17, $t18
100    add $t0, $t19, $t20
101    add $t0, $t21, $t22
102    add $t0, $t23, $t24
103    add $t0, $t25, $t26
104    add $t0, $t27, $t28
105    add $t0, $t29, $t30
106
107    add $t0, $t1, $t2
108    add $t0, $t3, $t4
109    add $t0, $t5, $t6
110    add $t0, $t7, $t8
111    add $t0, $t9, $t10
112    add $t0, $t11, $t12
113    add $t0, $t13, $t14
114    add $t0, $t15, $t16
115    add $t0, $t17, $t18
116    add $t0, $t19, $t20
117    add $t0, $t21, $t22
118    add $t0, $t23, $t24
119    add $t0, $t25, $t26
120    add $t0, $t27, $t28
121    add $t0, $t29, $t30
122
123    add $t0, $t1, $t2
124    add $t0, $t3, $t4
125    add $t0, $t5, $t6
126    add $t0, $t7, $t8
127    add $t0, $t9, $t10
128    add $t0, $t11, $t12
129    add $t0, $t13, $t14
130    add $t0, $t15, $t16
131    add $t0, $t17, $t18
132    add $t0, $t19, $t20
133    add $t0, $t21, $t22
134    add $t0, $t23, $t24
135    add $t0, $t25, $t26
136    add $t0, $t27, $t28
137    add $t0, $t29, $t30
138
139    add $t0, $t1, $t2
140    add $t0, $t3, $t4
141    add $t0, $t5, $t6
142    add $t0, $t7, $t8
143    add $t0, $t9, $t10
144    add $t0, $t11, $t12
145    add $t0, $t13, $t14
146    add $t0, $t15, $t16
147    add $t0, $t17, $t18
148    add $t0, $t19, $t20
149    add $t0, $t21, $t22
150    add $t0, $t23, $t24
151    add $t0, $t25, $t26
152    add $t0, $t27, $t28
153    add $t0, $t29, $t30
154
155    add $t0, $t1, $t2
156    add $t0, $t3, $t4
157    add $t0, $t5, $t6
158    add $t0, $t7, $t8
159    add $t0, $t9, $t10
160    add $t0, $t11, $t12
161    add $t0, $t13, $t14
162    add $t0, $t15, $t16
163    add $t0, $t17, $t18
164    add $t0, $t19, $t20
165    add $t0, $t21, $t22
166    add $t0, $t23, $t24
167    add $t0, $t25, $t26
168    add $t0, $t27, $t28
169    add $t0, $t29, $t30
170
171    add $t0, $t1, $t2
172    add $t0, $t3, $t4
173    add $t0, $t5, $t6
174    add $t0, $t7, $t8
175    add $t0, $t9, $t10
176    add $t0, $t11, $t12
177    add $t0, $t13, $t14
178    add $t0, $t15, $t16
179    add $t0, $t17, $t18
180    add $t0, $t19, $t20
181    add $t0, $t21, $t22
182    add $t0, $t23, $t24
183    add $t0, $t25, $t26
184    add $t0, $t27, $t28
185    add $t0, $t29, $t30
186
187    add $t0, $t1, $t2
188    add $t0, $t3, $t4
189    add $t0, $t5, $t6
190    add $t0, $t7, $t8
191    add $t0, $t9, $t10
192    add $t0, $t11, $t12
193    add $t0, $t13, $t14
194    add $t0, $t15, $t16
195    add $t0, $t17, $t18
196    add $t0, $t19, $t20
197    add $t0, $t21, $t22
198    add $t0, $t23, $t24
199    add $t0, $t25, $t26
200    add $t0, $t27, $t28
201    add $t0, $t29, $t30
202
203    add $t0, $t1, $t2
204    add $t0, $t3, $t4
205    add $t0, $t5, $t6
206    add $t0, $t7, $t8
207    add $t0, $t9, $t10
208    add $t0, $t11, $t12
209    add $t0, $t13, $t14
210    add $t0, $t15, $t16
211    add $t0, $t17, $t18
212    add $t0, $t19, $t20
213    add $t0, $t21, $t22
214    add $t0, $t23, $t24
215    add $t0, $t25, $t26
216    add $t0, $t27, $t28
217    add $t0, $t29, $t30
218
219    add $t0, $t1, $t2
220    add $t0, $t3, $t4
221    add $t0, $t5, $t6
222    add $t0, $t7, $t8
223    add $t0, $t9, $t10
224    add $t0, $t11, $t12
225    add $t0, $t13, $t14
226    add $t0, $t15, $t16
227    add $t0, $t17, $t18
228    add $t0, $t19, $t20
229    add $t0, $t21, $t22
230    add $t0, $t23, $t24
231    add $t0, $t25, $t26
232    add $t0, $t27, $t28
233    add $t0, $t29, $t30
234
235    add $t0, $t1, $t2
236    add $t0, $t3, $t4
237    add $t0, $t5, $t6
238    add $t0, $t7, $t8
239    add $t0, $t9, $t10
240    add $t0, $t11, $t12
241    add $t0, $t13, $t14
242    add $t0, $t15, $t16
243    add $t0, $t17, $t18
244    add $t0, $t19, $t20
245    add $t0, $t21, $t22
246    add $t0, $t23, $t24
247    add $t0, $t25, $t26
248    add $t0, $t27, $t28
249    add $t0, $t29, $t30
250
251    add $t0, $t1, $t2
252    add $t0, $t3, $t4
253    add $t0, $t5, $t6
254    add $t0, $t7, $t8
255    add $t0, $t9, $t10
256    add $t0, $t11, $t12
257    add $t0, $t13, $t14
258    add $t0, $t15, $t16
259    add $t0, $t17, $t18
260    add $t0, $t19, $t20
261    add $t0, $t21, $t22
262    add $t0, $t23, $t24
263    add $t0, $t25, $t26
264    add $t0, $t27, $t28
265    add $t0, $t29, $t30
266
267    add $t0, $t1, $t2
268    add $t0, $t3, $t4
269    add $t0, $t5, $t6
270    add $t0, $t7, $t8
271    add $t0, $t9, $t10
272    add $t0, $t11, $t12
273    add $t0, $t13, $t14
274    add $t0, $t15, $t16
275    add $t0, $t17, $t18
276    add $t0, $t19, $t20
277    add $t0, $t21, $t22
278    add $t0, $t23, $t24
279    add $t0, $t25, $t26
280    add $t0, $t27, $t28
281    add $t0, $t29, $t30
282
283    add $t0, $t1, $t2
284    add $t0, $t3, $t4
285    add $t0, $t5, $t6
286    add $t0, $t7, $t8
287    add $t0, $t9, $t10
288    add $t0, $t11, $t12
289    add $t0, $t13, $t14
290    add $t0, $t15, $t16
291    add $t0, $t17, $t18
292    add $t0, $t19, $t20
293    add $t0, $t21, $t22
294    add $t0, $t23, $t24
295    add $t0, $t25, $t26
296    add $t0, $t27, $t28
297    add $t0, $t29, $t30
298
299    add $t0, $t1, $t2
300    add $t0, $t3, $t4
301    add $t0, $t5, $t6
302    add $t0, $t7, $t8
303    add $t0, $t9, $t10
304    add $t0, $t11, $t12
305    add $t0, $t13, $t14
306    add $t0, $t15, $t16
307    add $t0, $t17, $t18
308    add $t0, $t19, $t20
309    add $t0, $t21, $t22
310    add $t0, $t23, $t24
311    add $t0, $t25, $t26
312    add $t0, $t27, $t28
313    add $t0, $t29, $t30
314
315    add $t0, $t1, $t2
316    add $t0, $t3, $t4
317    add $t0, $t5, $t6
318    add $t0, $t7, $t8
319    add $t0, $t9, $t10
320    add $t0, $t11, $t12
321    add $t0, $t13, $t14
322    add $t0, $t15, $t16
323    add $t0, $t17, $t18
324    add $t0, $t19, $t20
325    add $t0, $t21, $t22
326    add $t0, $t23, $t24
327    add $t0, $t25, $t26
328    add $t0, $t27, $t28
329    add $t0, $t29, $t30
330
331    add $t0, $t1, $t2
332    add $t0, $t3, $t4
333    add $t0, $t5, $t6
334    add $t0, $t7, $t8
335    add $t0, $t9, $t10
336    add $t0, $t11, $t12
337    add $t0, $t13, $t14
338    add $t0, $t15, $t16
339    add $t0, $t17, $t18
340    add $t0, $t19, $t20
341    add $t0, $t21, $t22
342    add $t0, $t23, $t24
343    add $t0, $t25, $t26
344    add $t0, $t27, $t28
345    add $t0, $t29, $t30
346
347    add $t0, $t1, $t2
348    add $t0, $t3, $t4
349    add $t0, $t5, $t6
350    add $t0, $t7, $t8
351    add $t0, $t9, $t10
352    add $t0, $t11, $t12
353    add $t0, $t13, $t14
354    add $t0, $t15, $t16
355    add $t0, $t17, $t18
356    add $t0, $t19, $t20
357    add $t0, $t21, $t22
358    add $t0, $t23, $t24
359    add $t0, $t25, $t26
360    add $t0, $t27, $t28
361    add $t0, $t29, $t30
362
363    add $t0, $t1, $t2
364    add $t0, $t3, $t4
365    add $t0, $t5, $t6
366    add $t0, $t7, $t8
367    add $t0, $t9, $t10
368    add $t0, $t11, $t12
369    add $t0, $t13, $t14
370    add $t0, $t15, $t16
371    add $t0, $t17, $t18
372    add $t0, $t19, $t20
373    add $t0, $t21, $t22
374    add $t0, $t23, $t24
375    add $t0, $t25, $t26
376    add $t0, $t27, $t28
377    add $t0, $t29, $t30
378
379    add $t0, $t1, $t2
380    add $t0, $t3, $t4
381    add $t0, $t5, $t6
382    add $t0, $t7, $t8
383    add $t0, $t9, $t10
384    add $t0, $t11, $t12
385    add $t0, $t13, $t14
386    add $t0, $t15, $t16
387    add $t0, $t17, $t18
388    add $t0, $t19, $t20
389    add $t0, $t21, $t22
390    add $t0, $t23, $t24
391    add $t0, $t25, $t26
392    add $t0, $t27, $t28
393    add $t0, $t29, $t30
394
395    add $t0, $t1, $t2
396    add $t0, $t3, $t4
397    add $t0, $t5, $t6
398    add $t0, $t7, $t8
399    add $t0, $t9, $t10
400    add $t0, $t11, $t12
401    add $t0, $t13, $t14
402    add $t0, $t15, $t16
403    add $t0, $t17, $t18
404    add $t0, $t19, $t20
405    add $t0, $t21, $t22
406    add $t0, $t23, $t24
407    add $t0, $t25, $t26
408    add $t0, $t27, $t28
409    add $t0, $t29, $t30
410
411    add $t0, $t1, $t2
412    add $t0, $t3, $t4
413    add $t0, $t5, $t6
414    add $t0, $t7, $t8
415    add $t0, $t9, $t10
416    add $t0, $t11, $t12
417    add $t0, $t13, $t14
418    add $t0, $t15, $t16
419    add $t0, $t17, $t18
420    add $t0, $t19, $t20
421    add $t0, $t21, $t22
422    add $t0, $t23, $t24
423    add $t0, $t25, $t26
424    add $t0, $t27, $t28
425    add $t0, $t29, $t30
426
427    add $t0, $t1, $t2
428    add $t0, $t3, $t4
429    add $t0, $t5, $t6
430    add $t0, $t7, $t8
431    add $t0, $t9, $t10
432    add $t0, $t11, $t12
433    add $t0, $t13, $t14
434    add $t0, $t15, $t16
435    add $t0, $t17, $t18
436    add $t0, $t19, $t20
437    add $t0, $t21, $t22
438    add $t0, $t23, $t24
439    add $t0, $t25, $t26
440    add $t0, $t27, $t28
441    add $t0, $t29, $t30
442
443    add $t0, $t1, $t2
444    add $t0, $t3, $t4
445    add $t0, $t5, $t6
446    add $t0, $t7, $t8
447    add $t0, $t9, $t10
448    add $t0, $t11, $t12
449    add $t0, $t13, $t14
450    add $t0, $t15, $t16
451    add $t0, $t17, $t18
452    add $t0, $t19, $t20
453    add $t0, $t21, $t22
454    add $t0, $t23, $t24
455    add $t0, $t25, $t26
456    add $t0, $t27, $t28
457    add $t0, $t29, $t30
458
459    add $t0, $t1, $t2
460    add $t0, $t3, $t4
461    add $t0, $t5, $t6
462    add $t0, $t7, $t8
463    add $t0, $t9, $t10
464    add $t0, $t11, $t12
465    add $t0, $t13, $t14
466    add $t0, $t15, $t16
467    add $t0, $t17, $t18
468    add $t0, $t19, $t20
469    add $t0, $t21, $t22
470    add $t0, $t23, $t24
471    add $t0, $t25, $t26
472    add $t0, $t27, $t28
473    add $t0, $t29, $t30
474
475    add $t0, $t1, $t2
476    add $t0, $t3, $t4
477    add $t0, $t5, $t6
478    add $t0, $t7, $t8
479    add $t0, $t9, $t10
480    add $t0, $t11, $t12
481    add $t0, $t13, $t14
482    add $t0, $t15, $t16
483    add $t0, $t17, $t18
484    add $t0, $t19, $t20
485    add $t0, $t21, $t22
486    add $t0, $t23, $t24
487    add $t0, $t25, $t26
488    add $t0, $t27, $t28
489    add $t0, $t29, $t30
490
491    add $t0, $t1, $t2
492    add $t0, $t3, $t4
493    add $t0, $t5, $t6
494    add $t0, $t7, $t8
495    add $t0, $t9, $t10
496    add $t0, $t11, $t12
497    add $t0, $t13, $t14
498    add $t0, $t15, $t16
499    add $t0, $t17, $t18
500    add $t0, $t19, $t20
501    add $t0, $t21, $t22
502    add $t0, $t23, $t24
503    add $t0, $t25, $t26
504    add $t0, $t27, $t28
505    add $t0, $t29, $t30
506
507    add $t0, $t1, $t2
508    add $t0, $t3, $t4
509    add $t0, $t5, $t6
510    add $t0, $t7, $t8
511    add $t0, $t9, $t10
512    add $t0, $t11, $t12
513    add $t0, $t13, $t14
514    add $t0, $t15, $t16
515    add $t0, $t17, $t18
516    add $t0, $t19, $t20
517    add $t0, $t21, $t22
518    add $t0, $t23, $t24
519    add $t0, $t25, $t26
520    add $t0, $t27, $t28
521    add $t0, $t29, $t30
522
523    add $t0, $t1, $t2
524    add $t0, $t3, $t4
525    add $t0, $t5, $t6
526    add $t0, $t7, $t8
527    add $t0, $t9, $t10
528    add $t0, $t11, $t12
529    add $t0, $t13, $t14
530    add $t0, $t15, $t16
531    add $t0, $t17, $t18
532    add $t0, $t19, $t20
533    add $t0, $t21, $t22
534    add $t0, $t23, $t24
535    add $t0, $t25, $t26
536    add $t0, $t27, $t28
537    add $t0, $t29, $t30
538
539    add $t0, $t1, $t2
540    add $t0, $t3, $t4
541    add $t0, $t5, $t6
542    add $t0, $t7, $t8
543    add $t0, $t9, $t10
544    add $t0, $t11, $t12
545    add $t0, $t13, $t14
546    add $t0, $t15, $t16
547    add $t0, $t17, $t18
548    add $t0, $t19, $t20
549    add $t0, $t21, $t22
550    add $t0, $t23, $t24
551    add $t0, $t25, $t26
552    add $t0, $t27, $t28
553    add $t0, $t29, $t30
554
555    add $t0, $t1, $t2
556    add $t0, $t3, $t4
557    add $t0, $t5, $t6
558    add $t0, $t7, $t8
559    add $t0, $t9, $t10
560    add $t0, $t11, $t12
561    add $t0, $t13, $t14
562    add $t0, $t15, $t16
563    add $t0, $t17, $t18
564    add $t0, $t19, $t20
565    add $t0, $t21, $t22
566    add $t0, $t23, $t24
567    add $t0, $t25, $t26
568    add $t0, $t27, $t28
569    add $t0, $t29, $t30
570
571    add $t0, $t1, $t2
572    add $t0, $t3, $t4
573    add $t0, $t5, $t6
574    add $t0, $t7, $t8
575    add $t0, $t9, $t10
576    add $t0, $t11, $t12
577    add $t0, $t13, $t14
578    add $t0, $t15, $t16
579    add $t0, $t17, $t18
580    add $t0, $t19, $t20
581    add $t0, $t21, $t22
582    add $t0, $t23, $t24
583    add $t0, $t25, $t26
584    add $t0, $t27, $t28
585    add $t0, $t29, $t30
586
587    add $t0, $t1, $t2
588    add $t0, $t3, $t4
589    add $t0, $t5, $t6
590    add $t0, $t7, $t8
591    add $t0, $t9, $t10
592    add $t0, $t11, $t12
593    add $t0, $t13, $t14
594    add $t0, $t15, $t16
595    add $t0, $t17, $t18
596    add $t0, $t19, $t20
597    add $t0, $t21, $t22
598    add $t0, $t23, $t24
599    add $t0, $t25, $t26
600    add $t0, $t27, $t28
601    add $t0, $t29, $t30
602
603    add $t0, $t1, $t2
604    add $t0, $t3, $t4
605    add $t0, $t5, $t6
606    add $t0, $t7, $t8
607    add $t0, $t9, $t10
608    add $t0, $t11, $t12
609    add $t0, $t13, $t14
610    add $t0, $t15, $t16
611    add $t0, $t17, $t18
612    add $t0, $t19, $t20
613    add $t0, $t21, $t22
614    add $t0, $t23, $t24
615    add $t0, $t25, $t26
616    add $t0, $t27, $t28
617    add $t0, $t29, $t30
618
619    add $t0, $t1, $t2
620    add $t0, $t3, $t4
621    add $t0, $t5, $t6
622    add $t0, $t7, $t8
623    add $t0, $t9, $t10
624    add $t0, $t11, $t12
625    add $t0, $t13, $t14
626    add $t0, $t15, $t16
627    add $t0, $t17, $t18
628    add $t0, $t19, $t20
629    add $t0, $t21, $t22
630    add $t0, $t23, $t24
631    add $t0, $t25, $t26
632    add $t0, $t27, $t28
633    add $t0, $t29, $t30
634
635    add $t0, $t1, $t2
636    add $t0, $t3, $t4
637    add $t0, $t5, $t6
638    add $t0, $t7, $t8
639    add $t0, $t9, $t10
640    add $t0, $t11, $t12
641    add $t0, $t13, $t14
642    add $t0, $t15, $t16
643    add $t0, $t17, $t18
644    add $t0, $t19, $t20
645    add $t0, $t21, $t22
646    add $t0, $t23, $t24
647    add $t0, $t25, $t26
648    add $t0, $t27, $t28
649    add $t0, $t29, $t30
650
651    add $t0, $t1, $t2
652    add $t0, $t3, $t4
653    add $t0, $t5, $t6
654    add $t0, $t7, $t8
655    add $t0, $t9, $t10
656    add $t0, $t11, $t12
657    add $t0, $t13, $t14
658    add $t0, $t15, $t16
659    add $t0, $t17, $t18
660    add $t0, $t19, $t20
661    add $t0, $t21, $t22
662    add $t0, $t23, $t24
663    add $t0, $t25, $t26
664    add $t0, $t27, $t28
665    add $t0, $t29, $t30
666
667    add $t0, $t1, $t2
668    add $t0, $t3, $t4
669    add $t0, $t5, $t6
670    add $t0, $t7, $t8
671    add $t0, $t9, $t10
672    add $t0, $t11, $t12
673    add $t0, $t13, $t14
674    add $t0, $t15, $t16
675    add $t0, $t17, $t18
676    add $t0, $t19, $t20
677    add $t0, $t21, $t22
678    add $t0, $t23, $t24
679    add $t0, $t25, $t26
680    add $t0, $t27, $t28
681    add $t0, $t29, $t30
682
683    add $t0, $t1, $t2
684    add $t0, $t3, $t4
685    add $t0, $t5, $t6
686    add $t0, $t7, $t8
687    add $t0, $t9, $t10
688    add $t0, $t11, $t12
689    add $t0, $t13, $t14
690    add $t0, $t15, $t16
691    add $t0, $t17, $t18
692    add $t0, $t19, $t20
693    add $t0, $t21, $t22
694    add $t0, $t23, $t24
695    add $t0, $t25, $t26
696    add $t0, $t27, $t28
697    add $t0, $t29, $t30
698
699    add $t0, $t1, $t2
700    add $t0, $t3, $t4
701    add $t0, $t5, $t6
702    add $t0, $t7, $t8
703    add $t0, $t9, $t10
704    add $t0, $t11, $t12
705    add $t0, $t13, $t14
706    add $t0, $t15, $t16
707    add $t0, $t17, $t18
708    add $t0, $t19, $t20
709    add $t0, $t21, $t22
710
```

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 (Basic form: OP rd, rs, rt)
 - **I Instructions:** instructions that also use intermediate values.
addi \$s1, \$s2, 100

MIPS Commands

The screenshot shows a MIPS assembly debugger interface. The top menu bar includes 'File', 'Edit', 'Get', 'Show/Hide Demo', 'User Guide | Unit Tests | Docs', 'Assembly', 'Data', 'Run', 'Break', 'Stack View', 'Hello World', 'Code Gen Data String', 'Interactive', 'Decimal Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with buttons for 'Step', 'Run', 'Break', 'Create auto watching', and 'Stop'.

The assembly code window contains the following instructions:

```
1 # Shows 'Hello world' at the top of the stack
2 .text
3 .globl _start
4 .data
5 _start: .asciz "Hello world\n"
6 .text
7 _start: li $t0, 111901
8 sb $t0, 111901
9 li $t1, 111902
10 sb $t1, 111902
11 li $t2, 111903
12 sb $t2, 111903
13 li $t3, 111904
14 sb $t3, 111904
15 li $t4, 111905
16 sb $t4, 111905
17 li $t5, 111906
18 sb $t5, 111906
19 li $t6, 111907
20 sb $t6, 111907
21 li $t7, 111908
22 sb $t7, 111908
23 li $t8, 111909
24 sb $t8, 111909
25 li $t9, 111910
26 sb $t9, 111910
27 li $t10, 111911
28 sb $t10, 111911
29 li $t11, 111912
30 sb $t11, 111912
31 li $t12, 111913
32 sb $t12, 111913
33 li $t13, 111914
34 sb $t13, 111914
35 li $t14, 111915
36 sb $t14, 111915
37 li $t15, 111916
38 sb $t15, 111916
39 li $t16, 111917
40 sb $t16, 111917
41 li $t17, 111918
42 sb $t17, 111918
43 li $t18, 111919
44 sb $t18, 111919
45 li $t19, 111920
46 sb $t19, 111920
47 li $t20, 111921
48 sb $t20, 111921
49 li $t21, 111922
50 sb $t21, 111922
51 li $t22, 111923
52 sb $t22, 111923
53 li $t23, 111924
54 sb $t23, 111924
55 li $t24, 111925
56 sb $t24, 111925
57 li $t25, 111926
58 sb $t25, 111926
59 li $t26, 111927
60 sb $t26, 111927
61 li $t27, 111928
62 sb $t27, 111928
63 li $t28, 111929
64 sb $t28, 111929
65 li $t29, 111930
66 sb $t29, 111930
67 li $t30, 111931
68 sb $t30, 111931
69 li $t31, 111932
70 sb $t31, 111932
71 li $t32, 111933
72 sb $t32, 111933
73 li $t33, 111934
74 sb $t33, 111934
75 li $t34, 111935
76 sb $t34, 111935
77 li $t35, 111936
78 sb $t35, 111936
79 li $t36, 111937
80 sb $t36, 111937
81 li $t37, 111938
82 sb $t37, 111938
83 li $t38, 111939
84 sb $t38, 111939
85 li $t39, 111940
86 sb $t39, 111940
87 li $t40, 111941
88 sb $t40, 111941
89 li $t41, 111942
90 sb $t41, 111942
91 li $t42, 111943
92 sb $t42, 111943
93 li $t43, 111944
94 sb $t43, 111944
95 li $t44, 111945
96 sb $t44, 111945
97 li $t45, 111946
98 sb $t45, 111946
99 li $t46, 111947
100 sb $t46, 111947
101 li $t47, 111948
102 sb $t47, 111948
103 li $t48, 111949
104 sb $t48, 111949
105 li $t49, 111950
106 sb $t49, 111950
107 li $t50, 111951
108 sb $t50, 111951
109 li $t51, 111952
110 sb $t51, 111952
111 li $t52, 111953
112 sb $t52, 111953
113 li $t53, 111954
114 sb $t53, 111954
115 li $t54, 111955
116 sb $t54, 111955
117 li $t55, 111956
118 sb $t55, 111956
119 li $t56, 111957
120 sb $t56, 111957
121 li $t57, 111958
122 sb $t57, 111958
123 li $t58, 111959
124 sb $t58, 111959
125 li $t59, 111960
126 sb $t59, 111960
127 li $t60, 111961
128 sb $t60, 111961
129 li $t61, 111962
130 sb $t61, 111962
131 li $t62, 111963
132 sb $t62, 111963
133 li $t63, 111964
134 sb $t63, 111964
135 li $t64, 111965
136 sb $t64, 111965
137 li $t65, 111966
138 sb $t65, 111966
139 li $t66, 111967
140 sb $t66, 111967
141 li $t67, 111968
142 sb $t67, 111968
143 li $t68, 111969
144 sb $t68, 111969
145 li $t69, 111970
146 sb $t69, 111970
147 li $t70, 111971
148 sb $t70, 111971
149 li $t71, 111972
150 sb $t71, 111972
151 li $t72, 111973
152 sb $t72, 111973
153 li $t73, 111974
154 sb $t73, 111974
155 li $t74, 111975
156 sb $t74, 111975
157 li $t75, 111976
158 sb $t75, 111976
159 li $t76, 111977
160 sb $t76, 111977
161 li $t77, 111978
162 sb $t77, 111978
163 li $t78, 111979
164 sb $t78, 111979
165 li $t79, 111980
166 sb $t79, 111980
167 li $t80, 111981
168 sb $t80, 111981
169 li $t81, 111982
170 sb $t81, 111982
171 li $t82, 111983
172 sb $t82, 111983
173 li $t83, 111984
174 sb $t83, 111984
175 li $t84, 111985
176 sb $t84, 111985
177 li $t85, 111986
178 sb $t85, 111986
179 li $t86, 111987
180 sb $t86, 111987
181 li $t87, 111988
182 sb $t87, 111988
183 li $t88, 111989
184 sb $t88, 111989
185 li $t89, 111990
186 sb $t89, 111990
187 li $t90, 111991
188 sb $t90, 111991
189 li $t91, 111992
190 sb $t91, 111992
191 li $t92, 111993
192 sb $t92, 111993
193 li $t93, 111994
194 sb $t93, 111994
195 li $t94, 111995
196 sb $t94, 111995
197 li $t95, 111996
198 sb $t95, 111996
199 li $t96, 111997
200 sb $t96, 111997
201 li $t97, 111998
202 sb $t97, 111998
203 li $t98, 111999
204 sb $t98, 111999
205 li $t99, 111900
206 sb $t99, 111900
207 li $t100, 111901
208 sb $t100, 111901
209 li $t101, 111902
210 sb $t101, 111902
211 li $t102, 111903
212 sb $t102, 111903
213 li $t103, 111904
214 sb $t103, 111904
215 li $t104, 111905
216 sb $t104, 111905
217 li $t105, 111906
218 sb $t105, 111906
219 li $t106, 111907
220 sb $t106, 111907
221 li $t107, 111908
222 sb $t107, 111908
223 li $t108, 111909
224 sb $t108, 111909
225 li $t109, 111910
226 sb $t109, 111910
227 li $t110, 111911
228 sb $t110, 111911
229 li $t111, 111912
230 sb $t111, 111912
231 li $t112, 111913
232 sb $t112, 111913
233 li $t113, 111914
234 sb $t113, 111914
235 li $t114, 111915
236 sb $t114, 111915
237 li $t115, 111916
238 sb $t115, 111916
239 li $t116, 111917
240 sb $t116, 111917
241 li $t117, 111918
242 sb $t117, 111918
243 li $t118, 111919
244 sb $t118, 111919
245 li $t119, 111920
246 sb $t119, 111920
247 li $t120, 111921
248 sb $t120, 111921
249 li $t121, 111922
250 sb $t121, 111922
251 li $t122, 111923
252 sb $t122, 111923
253 li $t123, 111924
254 sb $t123, 111924
255 li $t124, 111925
256 sb $t124, 111925
257 li $t125, 111926
258 sb $t125, 111926
259 li $t126, 111927
260 sb $t126, 111927
261 li $t127, 111928
262 sb $t127, 111928
263 li $t128, 111929
264 sb $t128, 111929
265 li $t129, 111930
266 sb $t129, 111930
267 li $t130, 111931
268 sb $t130, 111931
269 li $t131, 111932
270 sb $t131, 111932
271 li $t132, 111933
272 sb $t132, 111933
273 li $t133, 111934
274 sb $t133, 111934
275 li $t134, 111935
276 sb $t134, 111935
277 li $t135, 111936
278 sb $t135, 111936
279 li $t136, 111937
280 sb $t136, 111937
281 li $t137, 111938
282 sb $t137, 111938
283 li $t138, 111939
284 sb $t138, 111939
285 li $t139, 111940
286 sb $t139, 111940
287 li $t140, 111941
288 sb $t140, 111941
289 li $t141, 111942
290 sb $t141, 111942
291 li $t142, 111943
292 sb $t142, 111943
293 li $t143, 111944
294 sb $t143, 111944
295 li $t144, 111945
296 sb $t144, 111945
297 li $t145, 111946
298 sb $t145, 111946
299 li $t146, 111947
300 sb $t146, 111947
301 li $t147, 111948
302 sb $t147, 111948
303 li $t148, 111949
304 sb $t148, 111949
305 li $t149, 111950
306 sb $t149, 111950
307 li $t150, 111951
308 sb $t150, 111951
309 li $t151, 111952
310 sb $t151, 111952
311 li $t152, 111953
312 sb $t152, 111953
313 li $t153, 111954
314 sb $t153, 111954
315 li $t154, 111955
316 sb $t154, 111955
317 li $t155, 111956
318 sb $t155, 111956
319 li $t156, 111957
320 sb $t156, 111957
321 li $t157, 111958
322 sb $t157, 111958
323 li $t158, 111959
324 sb $t158, 111959
325 li $t159, 111960
326 sb $t159, 111960
327 li $t160, 111961
328 sb $t160, 111961
329 li $t161, 111962
330 sb $t161, 111962
331 li $t162, 111963
332 sb $t162, 111963
333 li $t163, 111964
334 sb $t163, 111964
335 li $t164, 111965
336 sb $t164, 111965
337 li $t165, 111966
338 sb $t165, 111966
339 li $t166, 111967
340 sb $t166, 111967
341 li $t167, 111968
342 sb $t167, 111968
343 li $t168, 111969
344 sb $t168, 111969
345 li $t169, 111970
346 sb $t169, 111970
347 li $t170, 111971
348 sb $t170, 111971
349 li $t171, 111972
350 sb $t171, 111972
351 li $t172, 111973
352 sb $t172, 111973
353 li $t173, 111974
354 sb $t173, 111974
355 li $t174, 111975
356 sb $t174, 111975
357 li $t175, 111976
358 sb $t175, 111976
359 li $t176, 111977
360 sb $t176, 111977
361 li $t177, 111978
362 sb $t177, 111978
363 li $t178, 111979
364 sb $t178, 111979
365 li $t179, 111980
366 sb $t179, 111980
367 li $t180, 111981
368 sb $t180, 111981
369 li $t181, 111982
370 sb $t181, 111982
371 li $t182, 111983
372 sb $t182, 111983
373 li $t183, 111984
374 sb $t183, 111984
375 li $t184, 111985
376 sb $t184, 111985
377 li $t185, 111986
378 sb $t185, 111986
379 li $t186, 111987
380 sb $t186, 111987
381 li $t187, 111988
382 sb $t187, 111988
383 li $t188, 111989
384 sb $t188, 111989
385 li $t189, 111990
386 sb $t189, 111990
387 li $t190, 111991
388 sb $t190, 111991
389 li $t191, 111992
390 sb $t191, 111992
391 li $t192, 111993
392 sb $t192, 111993
393 li $t193, 111994
394 sb $t193, 111994
395 li $t194, 111995
396 sb $t194, 111995
397 li $t195, 111996
398 sb $t195, 111996
399 li $t196, 111997
400 sb $t196, 111997
401 li $t197, 111998
402 sb $t197, 111998
403 li $t198, 111999
404 sb $t198, 111999
405 li $t199, 111900
406 sb $t199, 111900
407 li $t200, 111901
408 sb $t200, 111901
409 li $t201, 111902
410 sb $t201, 111902
411 li $t202, 111903
412 sb $t202, 111903
413 li $t203, 111904
414 sb $t203, 111904
415 li $t204, 111905
416 sb $t204, 111905
417 li $t205, 111906
418 sb $t205, 111906
419 li $t206, 111907
420 sb $t206, 111907
421 li $t207, 111908
422 sb $t207, 111908
423 li $t208, 111909
424 sb $t208, 111909
425 li $t209, 111910
426 sb $t209, 111910
427 li $t210, 111911
428 sb $t210, 111911
429 li $t211, 111912
430 sb $t211, 111912
431 li $t212, 111913
432 sb $t212, 111913
433 li $t213, 111914
434 sb $t213, 111914
435 li $t214, 111915
436 sb $t214, 111915
437 li $t215, 111916
438 sb $t215, 111916
439 li $t216, 111917
440 sb $t216, 111917
441 li $t217, 111918
442 sb $t217, 111918
443 li $t218, 111919
444 sb $t218, 111919
445 li $t219, 111920
446 sb $t219, 111920
447 li $t220, 111921
448 sb $t220, 111921
449 li $t221, 111922
450 sb $t221, 111922
451 li $t222, 111923
452 sb $t222, 111923
453 li $t223, 111924
454 sb $t223, 111924
455 li $t224, 111925
456 sb $t224, 111925
457 li $t225, 111926
458 sb $t225, 111926
459 li $t226, 111927
460 sb $t226, 111927
461 li $t227, 111928
462 sb $t227, 111928
463 li $t228, 111929
464 sb $t228, 111929
465 li $t229, 111930
466 sb $t229, 111930
467 li $t230, 111931
468 sb $t230, 111931
469 li $t231, 111932
470 sb $t231, 111932
471 li $t232, 111933
472 sb $t232, 111933
473 li $t233, 111934
474 sb $t233, 111934
475 li $t234, 111935
476 sb $t234, 111935
477 li $t235, 111936
478 sb $t235, 111936
479 li $t236, 111937
480 sb $t236, 111937
481 li $t237, 111938
482 sb $t237, 111938
483 li $t238, 111939
484 sb $t238, 111939
485 li $t239, 111940
486 sb $t239, 111940
487 li $t240, 111941
488 sb $t240, 111941
489 li $t241, 111942
490 sb $t241, 111942
491 li $t242, 111943
492 sb $t242, 111943
493 li $t243, 111944
494 sb $t243, 111944
495 li $t244, 111945
496 sb $t244, 111945
497 li $t245, 111946
498 sb $t245, 111946
499 li $t246, 111947
500 sb $t246, 111947
501 li $t247, 111948
502 sb $t247, 111948
503 li $t248, 111949
504 sb $t248, 111949
505 li $t249, 111950
506 sb $t249, 111950
507 li $t250, 111951
508 sb $t250, 111951
509 li $t251, 111952
510 sb $t251, 111952
511 li $t252, 111953
512 sb $t252, 111953
513 li $t253, 111954
514 sb $t253, 111954
515 li $t254, 111955
516 sb $t254, 111955
517 li $t255, 111956
518 sb $t255, 111956
519 li $t256, 111957
520 sb $t256, 111957
521 li $t257, 111958
522 sb $t257, 111958
523 li $t258, 111959
524 sb $t258, 111959
525 li $t259, 111960
526 sb $t259, 111960
527 li $t260, 111961
528 sb $t260, 111961
529 li $t261, 111962
530 sb $t261, 111962
531 li $t262, 111963
532 sb $t262, 111963
533 li $t263, 111964
534 sb $t263, 111964
535 li $t264, 111965
536 sb $t264, 111965
537 li $t265, 111966
538 sb $t265, 111966
539 li $t266, 111967
540 sb $t266, 111967
541 li $t267, 111968
542 sb $t267, 111968
543 li $t268, 111969
544 sb $t268, 111969
545 li $t269, 111970
546 sb $t269, 111970
547 li $t270, 111971
548 sb $t270, 111971
549 li $t271, 111972
550 sb $t271, 111972
551 li $t272, 111973
552 sb $t272, 111973
553 li $t273, 111974
554 sb $t273, 111974
555 li $t274, 111975
556 sb $t274, 111975
557 li $t275, 111976
558 sb $t275, 111976
559 li $t276, 111977
560 sb $t276, 111977
561 li $t277, 111978
562 sb $t277, 111978
563 li $t278, 111979
564 sb $t278, 111979
565 li $t279, 111980
566 sb $t279, 111980
567 li $t280, 111981
568 sb $t280, 111981
569 li $t281, 111982
570 sb $t281, 111982
571 li $t282, 111983
572 sb $t282, 111983
573 li $t283, 111984
574 sb $t283, 111984
575 li $t284, 111985
576 sb $t284, 111985
577 li $t285, 111986
578 sb $t285, 111986
579 li $t286, 111987
580 sb $t286, 111987
581 li $t287, 111988
582 sb $t287, 111988
583 li $t288, 111989
584 sb $t288, 111989
585 li $t289, 111990
586 sb $t289, 111990
587 li $t290, 111991
588 sb $t290, 111991
589 li $t291, 111992
590 sb $t291, 111992
591 li $t292, 111993
592 sb $t292, 111993
593 li $t293, 111994
594 sb $t293, 111994
595 li $t294, 111995
596 sb $t294, 111995
597 li $t295, 111996
598 sb $t295, 111996
599 li $t296, 111997
600 sb $t296, 111997
601 li $t297, 111998
602 sb $t297, 111998
603 li $t298, 111999
604 sb $t298, 111999
605 li $t299, 111900
606 sb $t299, 111900
607 li $t300, 111901
608 sb $t300, 111901
609 li $t301, 111902
610 sb $t301, 111902
611 li $t302, 111903
612 sb $t302, 111903
613 li $t303, 111904
614 sb $t303, 111904
615 li $t304, 111905
616 sb $t304, 111905
617 li $t305, 111906
618 sb $t305, 111906
619 li $t306, 111907
620 sb $t306, 111907
621 li $t307, 111908
622 sb $t307, 111908
623 li $t308, 111909
624 sb $t308, 111909
625 li $t309, 111910
626 sb $t309, 111910
627 li $t310, 111911
628 sb $t310, 111911
629 li $t311, 111912
630 sb $t311, 111912
631 li $t312, 111913
632 sb $t312, 111913
633 li $t313, 111914
634 sb $t313, 111914
635 li $t314, 111915
636 sb $t314, 111915
637 li $t315, 111916
638 sb $t315, 111916
639 li $t316, 111917
640 sb $t316, 111917
641 li $t317, 111918
642 sb $t317, 111918
643 li $t318, 111919
644 sb $t318, 111919
645 li $t319, 111920
646 sb $t319, 111920
647 li $t320, 111921
648 sb $t320, 111921
649 li $t321, 111922
650 sb $t321, 111922
651 li $t322, 111923
652 sb $t322, 111923
653 li $t323, 111924
654 sb $t323, 111924
655 li $t324, 111925
656 sb $t324, 111925
657 li $t325, 111926
658 sb $t325, 111926
659 li $t326, 111927
660 sb $t326, 111927
661 li $t327, 111928
662 sb $t327, 111928
663 li $t328, 111929
664 sb $t328, 111929
665 li $t329, 111930
666 sb $t329, 111930
667 li $t330, 111931
668 sb $t330, 111931
669 li $t331, 111932
670 sb $t331, 111932
671 li $t332, 111933
672 sb $t332, 111933
673 li $t333, 111934
674 sb $t333, 111934
675 li $t334, 111935
676 sb $t334, 111935
677 li $t335, 111936
678 sb $t335, 111936
679 li $t336, 111937
680 sb $t336, 111937
681 li $t337, 111938
682 sb $t337, 111938
683 li $t338, 111939
684 sb $t338, 111939
685 li $t339, 111940
686 sb $t339, 111940
687 li $t340, 111941
688 sb $t340, 111941
689 li $t341, 111942
690 sb $t341, 111942
691 li $t342, 111943
692 sb $t342, 111943
693 li $t343, 111944
694 sb $t343, 111944
695 li $t344, 111945
696 sb $t344, 111945
697 li $t345, 111946
698 sb $t345, 
```

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 (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

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 (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)

Challenge:

Line: 3 Go! Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav 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 ADDI $sp, $sp, -13
3 ADDI $t0, $zero, 72 # H
4 SB $t0, 0($sp)
5 ADDI $t0, $zero, 101 # e
6 SB $t0, 1($sp)
7 ADDI $t0, $zero, 108 # l
8 SB $t0, 2($sp)
9 ADDI $t0, $zero, 108 # l
10 SB $t0, 3($sp)
11 ADDI $t0, $zero, 111 # o
12 SB $t0, 4($sp)
13 ADDI $t0, $zero, 32 # (space)
14 SB $t0, 5($sp)
15 ADDI $t0, $zero, 119 # w
16 SB $t0, 6($sp)
17 ADDI $t0, $zero, 111 # o
18 SB $t0, 7($sp)
19 ADDI $t0, $zero, 114 # r
20 SB $t0, 8($sp)
21 ADDI $t0, $zero, 108 # l
22 SB $t0, 9($sp)
23 ADDI $t0, $zero, 100 # d
24 SB $t0, 10($sp)
25 ADDI $t0, $zero, 33 # !
26 SB $t0, 11($sp)
27 ADDI $t0, $zero, 0 # (null)
28 SB $t0, 12($sp)
29
30 ADDI $v0, $zero, 4 # 4 is for print string
31 ADDI $a0, $sp, 0      # print to the log
32 syscall
```

Step Run Enable auto switching

S	T	A	V	Stack	Log
s0:	10				
s1:	9				
s2:	9				
s3:	22				
s4:	696				
s5:	976				
s6:	927				
s7:	418				

Write a program that prints out the alphabet: a b c d ... x y z

WeMIPS

(Demo with WeMIPS)

Today's Topics



- Design Patterns: Searching
- Python Recap
- Machine Language
- **Machine Language: Jumps & Loops**
- Binary & Hex Arithmetic

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.



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.



Jump Demo

Line: 18 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

```
1 ADDI $sp, $sp, -27      # Set up stack
2 ADDI $s3, $zero, 1       # Store 1 in a register
3 ADDI $t0, $zero, 97      # Set $t0 at 97 (a)
4 ADDI $s2, $zero, 26      # Use to test when you reach 26
5 SETUP: SB $t0, 0($sp)    # Next letter in $t0
6 ADDI $sp, $sp, 1         # Increment the stack
7 SUB $s2, $s2, $s3        # Decrease the counter by 1
8 ADDI $t0, $t0, 1         # Increment the letter
9 BEQ $s2, $zero, DONE     # Jump to done if $s2 == 0
10 J SETUP
11 J SETUP
12 DONE: ADDI $t0, $zero, 0 # Null (0) to terminate string
13 SB $t0, 0($sp)          # Add null to stack
14 ADDI $sp, $sp, -26      # Set up stack to print
15 ADDI $v0, $zero, 4       # 4 is for print string
16 ADDI $a0, $sp, 0         # Set $a0 to stack pointer
17 syscall                # Print to the log
```

(Demo
with
WeMIPS)

Step Run Enable auto switching

S T A V Stack Log

[Clear Log](#)

Emulation complete, returning to line 1

abcdefghijklmnopqrstuvwxyz

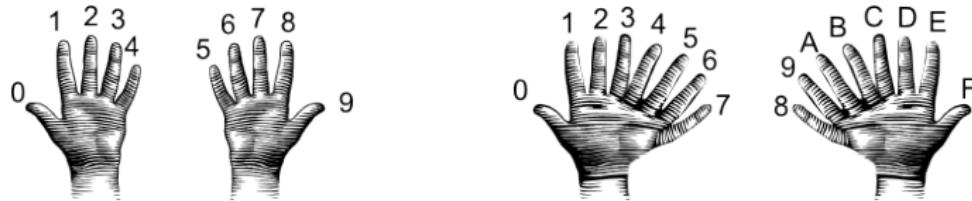


Today's Topics



- Design Patterns: Searching
- Python Recap
- Machine Language
- Machine Language: Jumps & Loops
- **Binary & Hex Arithmetic**

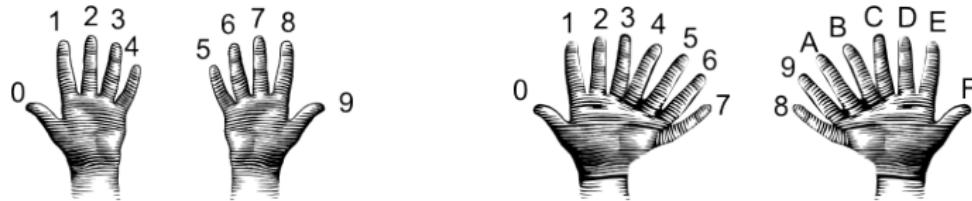
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - Convert first digit to decimal and multiple by 16.

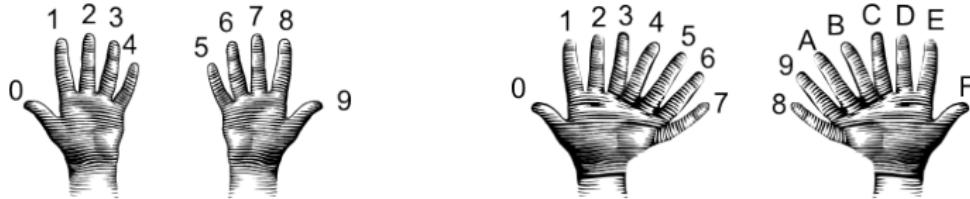
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - Convert first digit to decimal and multiple by 16.
 - Convert second digit to decimal and add to total.

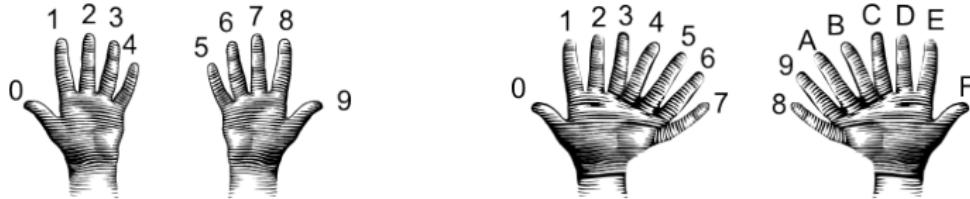
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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?

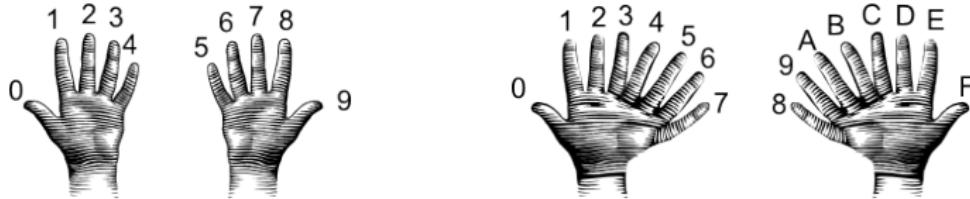
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

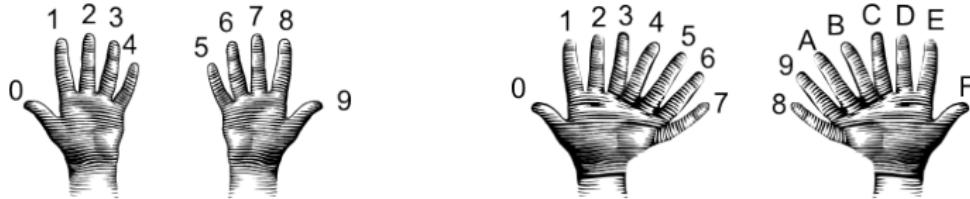
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

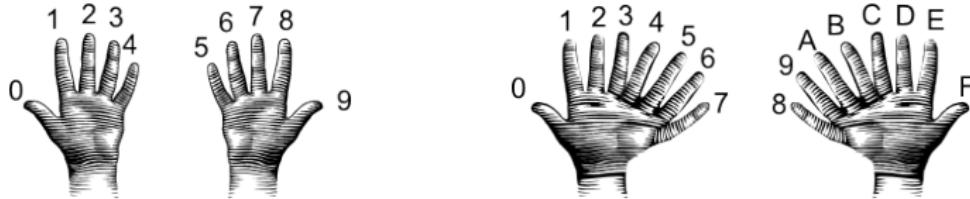
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):

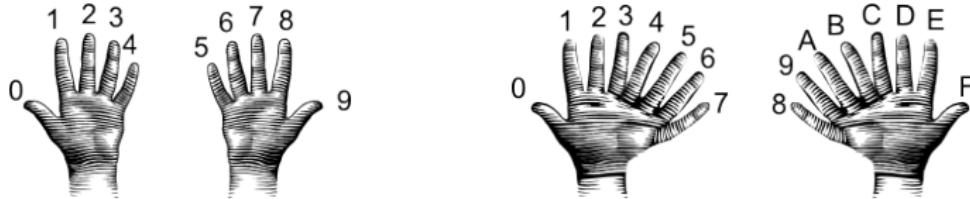
- Convert first digit to decimal and multiple by 16.
 - Convert second digit to decimal and add to total.
 - Example: what is 2A as a decimal number?

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

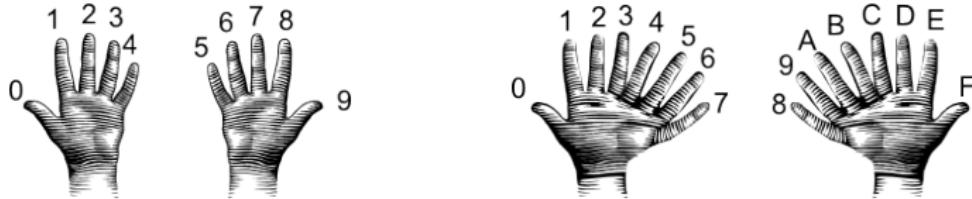
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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?

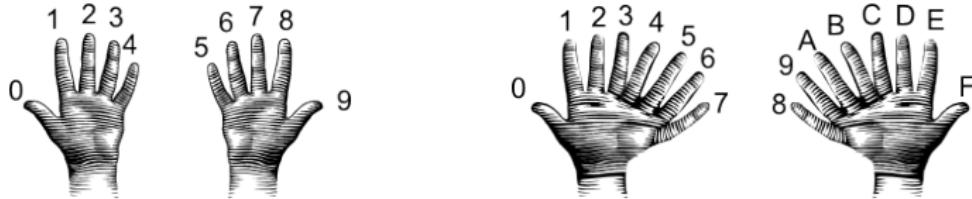
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

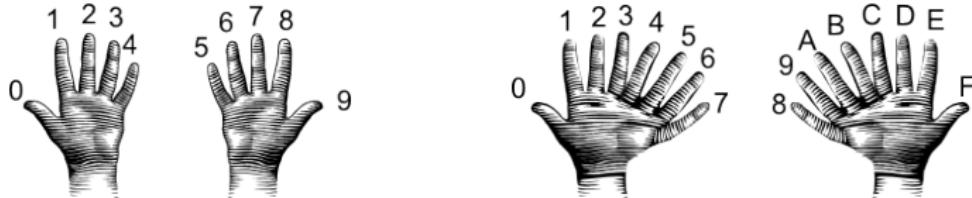
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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.

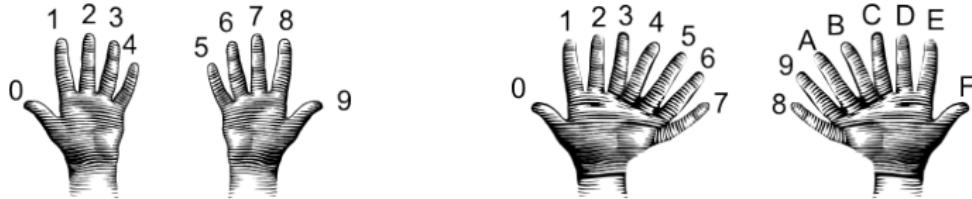
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
 - 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

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):

- 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.

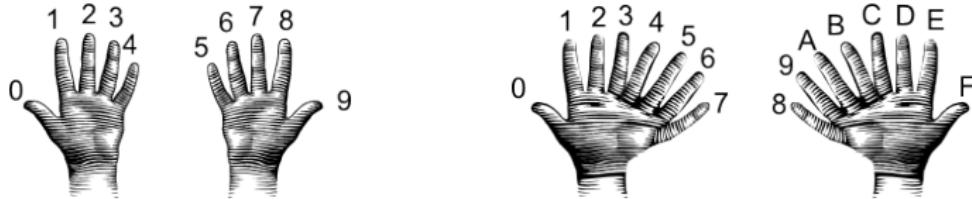
- Example: what is 99 as a decimal number?

9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):

- 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?

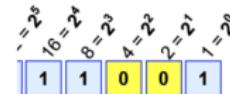
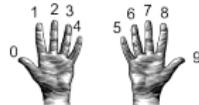
9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Answer is 153.

Decimal to Binary: Converting Between Bases

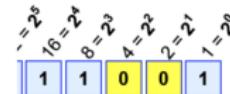
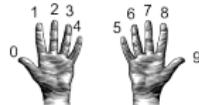


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

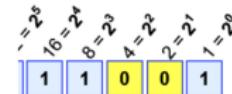


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

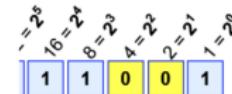
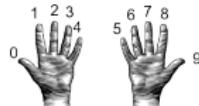


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

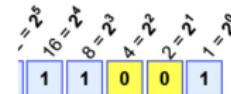
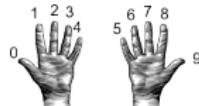


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

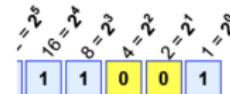


$$\text{Example: } 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Decimal to Binary: Converting Between Bases

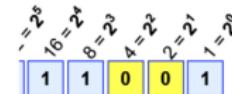
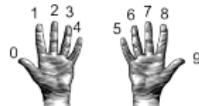


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

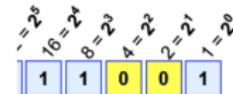
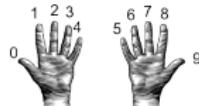


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Decimal to Binary: Converting Between Bases

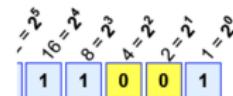
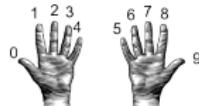


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$$

- 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.

Decimal to Binary: Converting Between Bases

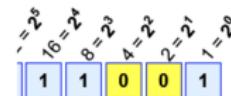
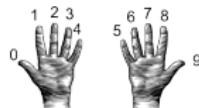


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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?

Decimal to Binary: Converting Between Bases



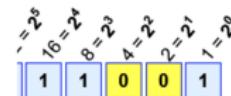
- From decimal to binary:

- 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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

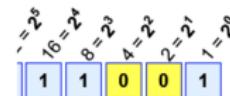
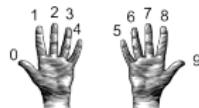
- From decimal to binary:

- 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:

Decimal to Binary: Converting Between Bases



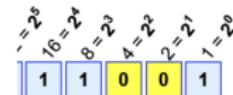
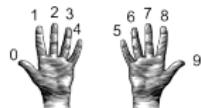
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

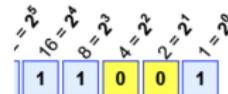
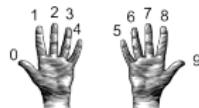
- From decimal to binary:

- 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:

Decimal to Binary: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$$

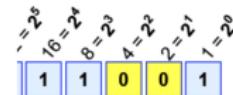
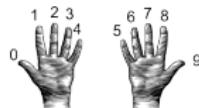
- From decimal to binary:

- 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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 = 16 + 8 + 4 = 28$

- From decimal to binary:

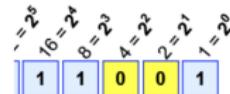
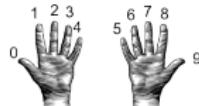
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

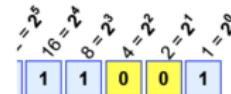
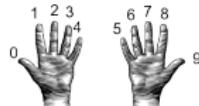
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

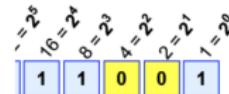
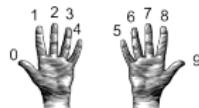
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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?

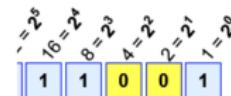
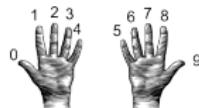
130/128 is 1 rem 2. First digit is 1: 1...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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?

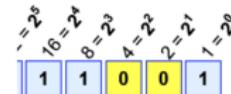
130/128 is 1 rem 2. First digit is 1: 1...

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:

Decimal to Binary: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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?

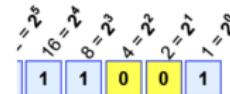
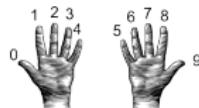
130/128 is 1 rem 2. First digit is 1: 1...

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...

Decimal to Binary: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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...

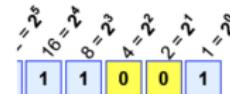
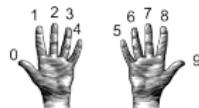
2/64 is 0 rem 2. Next digit is 0: 10...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 25$

- 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...

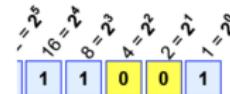
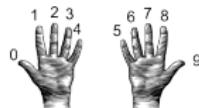
2/64 is 0 rem 2. Next digit is 0: 10...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

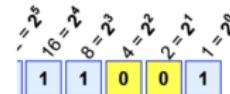
2/64 is 0 rem 2. Next digit is 0: 10...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

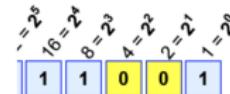
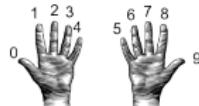
2/32 is 0 rem 2. Next digit is 0: 100...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

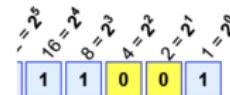
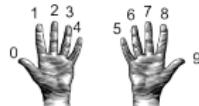
2/32 is 0 rem 2. Next digit is 0: 100...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

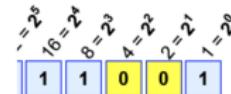
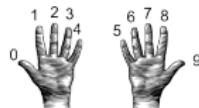
2/32 is 0 rem 2. Next digit is 0: 100...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

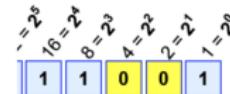
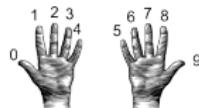
2/16 is 0 rem 2. Next digit is 0: 1000...

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.

Decimal to Binary: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$$

- 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...

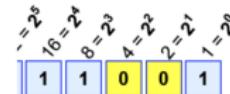
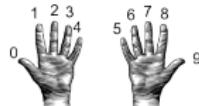
2/16 is 0 rem 2. Next digit is 0: 1000...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

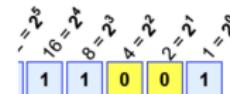
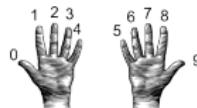
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

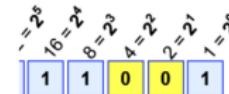
2/8 is 0 rem 2. Next digit is 0: 10000...

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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

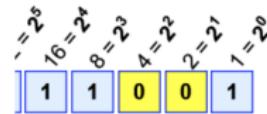
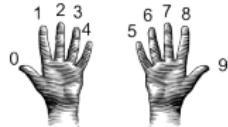
2/8 is 0 rem 2. Next digit is 0: 10000...

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

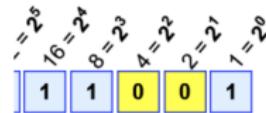
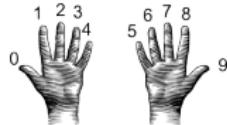
Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

Decimal to Binary: Converting Between Bases

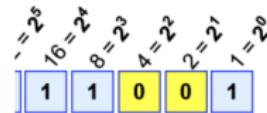


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99 / 128$ is 0 rem 99.

Decimal to Binary: Converting Between Bases

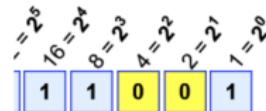


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases



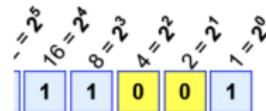
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

99/64 is 1 rem 35.

Decimal to Binary: Converting Between Bases



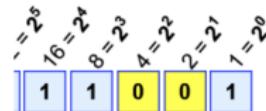
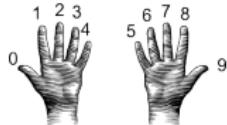
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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:

Decimal to Binary: Converting Between Bases



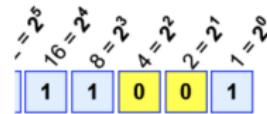
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

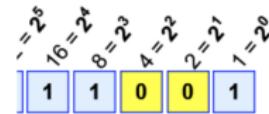
- Example: what is 99 in binary notation?

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

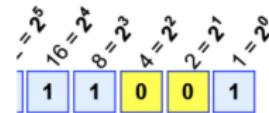
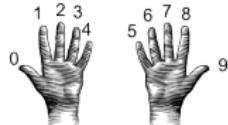
- Example: what is 99 in binary notation?

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

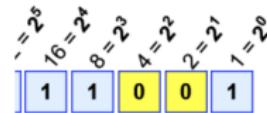
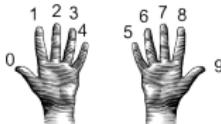
- Example: what is 99 in binary notation?

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- Example: what is 99 in binary notation?

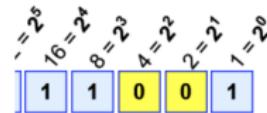
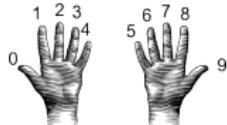
99/128 is 0 rem 99. First digit is 0: 0...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

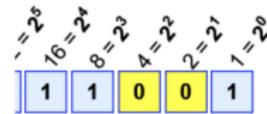
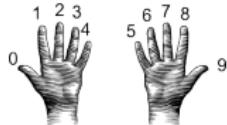
99/128 is 0 rem 99. First digit is 0: 0...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

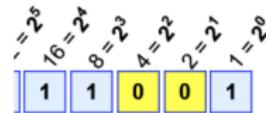
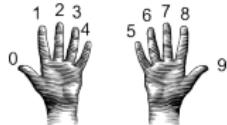
99/128 is 0 rem 99. First digit is 0: 0...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

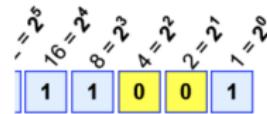
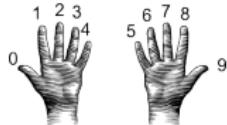
99/64 is 1 rem 35. Next digit is 1: 01...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

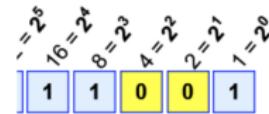
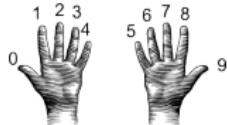
99/64 is 1 rem 35. Next digit is 1: 01...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

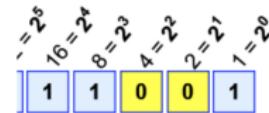
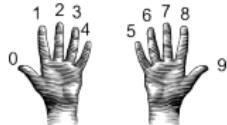
99/64 is 1 rem 35. Next digit is 1: 01...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

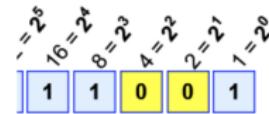
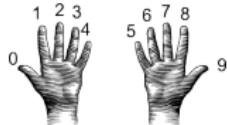
35/32 is 1 rem 3. Next digit is 1: 011...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

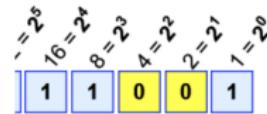
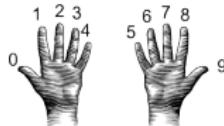
35/32 is 1 rem 3. Next digit is 1: 011...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

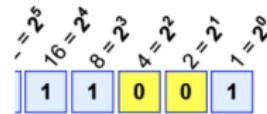
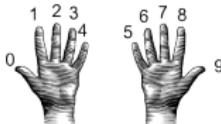
35/32 is 1 rem 3. Next digit is 1: 011...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

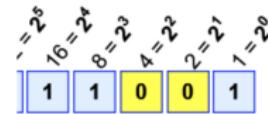
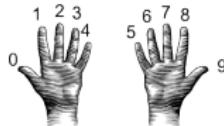
3/16 is 0 rem 3. Next digit is 0: 0110...

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.

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

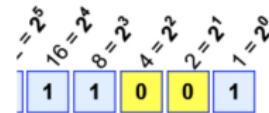
3/16 is 0 rem 3. Next digit is 0: 0110...

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:

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

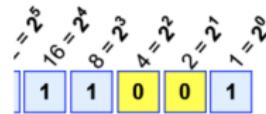
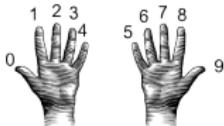
3/16 is 0 rem 3. Next digit is 0: 0110...

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...

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

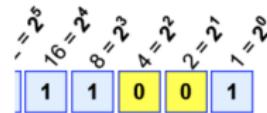
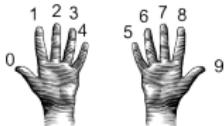
3/8 is 0 rem 3. Next digit is 0: 01100...

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

Decimal to Binary: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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...

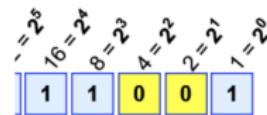
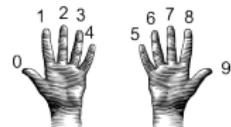
3/4 is 0 remainder 3. Next digit is 0: 011000...

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

Adding the last remainder: 01100011

Answer is 1100011.

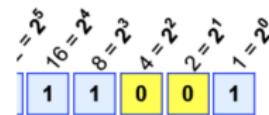
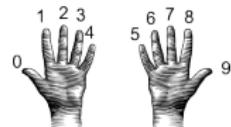
Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

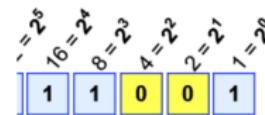
Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:
 - Set sum = last digit.
 - Multiply next digit by 2 = 2^1 . Add to sum.

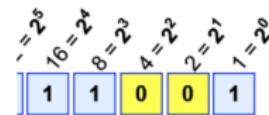
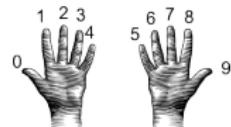
Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

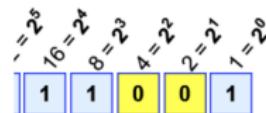
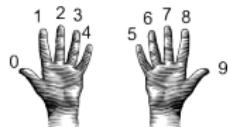
Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Binary to Decimal: Converting Between Bases

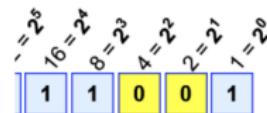
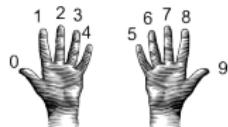


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Binary to Decimal: Converting Between Bases

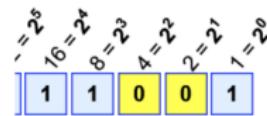
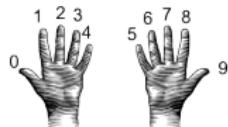


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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.

Binary to Decimal: Converting Between Bases

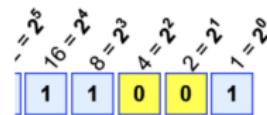
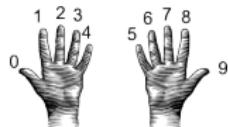


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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.

Binary to Decimal: Converting Between Bases

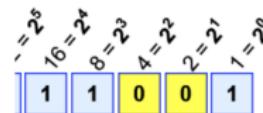
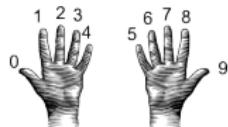


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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.

Binary to Decimal: Converting Between Bases

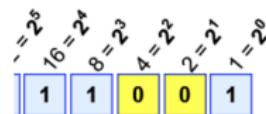
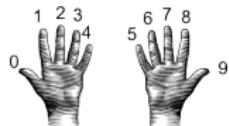


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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.

Binary to Decimal: Converting Between Bases



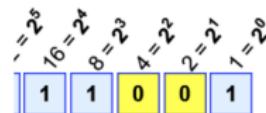
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- From binary to decimal:

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

Sum starts with:

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

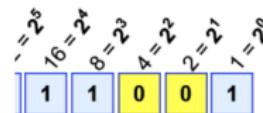
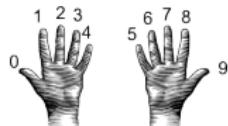
- From binary to decimal:

- 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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

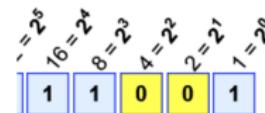
- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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

$0 * 2 = 0$. Add 0 to sum: 1

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

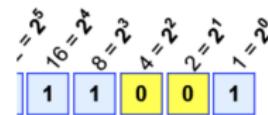
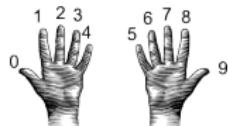
- Set sum = last digit.
- Multiply next digit by $2 = 2^1$. Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$.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

$0 * 2 = 0$. Add 0 to sum: 1

$1 * 4 = 4$. Add 4 to sum:

Binary to Decimal: Converting Between Bases



$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

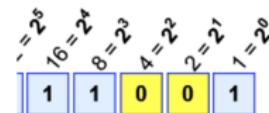
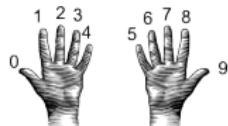
- Set sum = last digit.
- Multiply next digit by 2^1 . Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$. Add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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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?

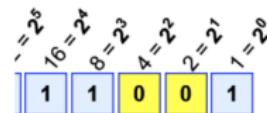
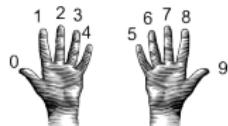
Sum starts with: 1

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

$1 \times 4 = 4$. Add 4 to sum: 5

$1 \times 8 = 8$. Add 8 to sum:

Binary to Decimal: Converting Between Bases



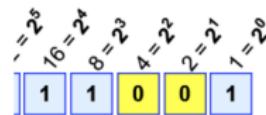
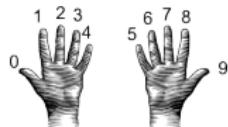
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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

Binary to Decimal: Converting Between Bases



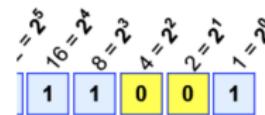
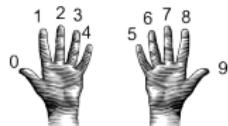
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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:

Binary to Decimal: Converting Between Bases



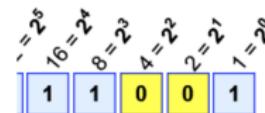
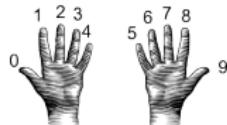
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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*2 = 0. Add 0 to sum: 1
1*4 = 4. Add 4 to sum: 5
1*8 = 8. Add 8 to sum: 13
1*16 = 16. Add 16 to sum: 29

Binary to Decimal: Converting Between Bases



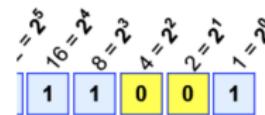
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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*2 = 0. Add 0 to sum: 1
1*4 = 4. Add 4 to sum: 5
1*8 = 8. Add 8 to sum: 13
1*16 = 16. Add 16 to sum: 29
1*32 = 32. Add 32 to sum:

Binary to Decimal: Converting Between Bases



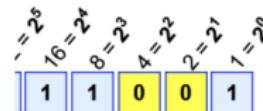
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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*2 = 0. Add 0 to sum: 1
1*4 = 4. Add 4 to sum: 5
1*8 = 8. Add 8 to sum: 13
1*16 = 16. Add 16 to sum: 29
1*32 = 32. Add 32 to sum: 61

Binary to Decimal: Converting Between Bases



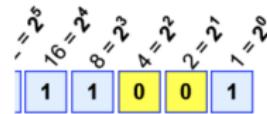
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 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*2 = 0. Add 0 to sum: 1
1*4 = 4. Add 4 to sum: 5
1*8 = 8. Add 8 to sum: 13
1*16 = 16. Add 16 to sum: 29
1*32 = 32. Add 32 to sum: 61

Binary to Decimal: Converting Between Bases

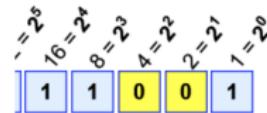
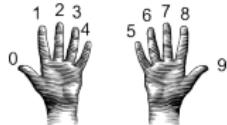


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 + 1 \times 2^0 = 16 + 8 + 4 + 0 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

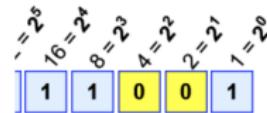
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

Binary to Decimal: Converting Between Bases



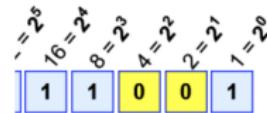
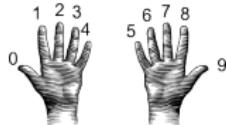
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

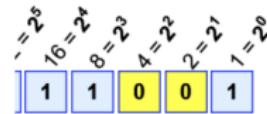
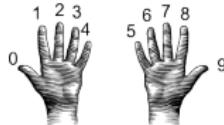
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

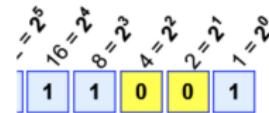
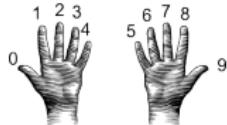
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

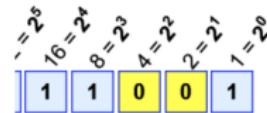
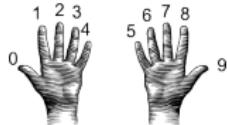
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

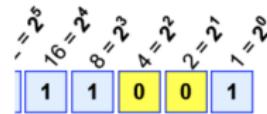
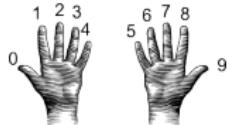
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

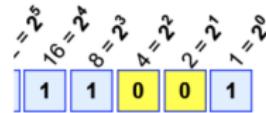
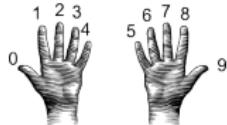
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

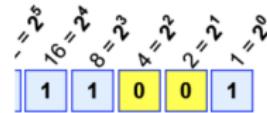
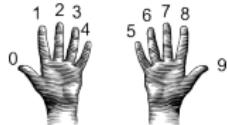
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

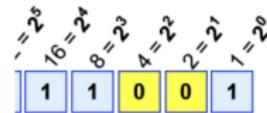
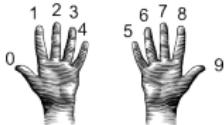
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases

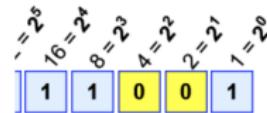
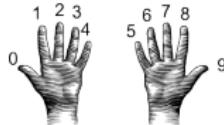


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 = 16 + 8 + 4 = 25$

- 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

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

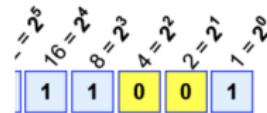
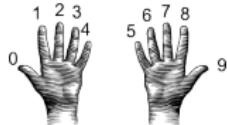
$0 \times 8 = 0$. Add 0 to sum: 4

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

$1 \times 32 = 32$. Add 32 to sum: 36

$0 \times 64 = 0$. Add 0 to sum:

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

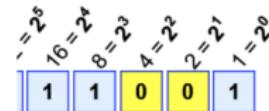
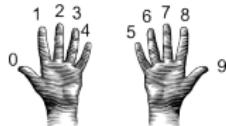
$0 \times 8 = 0$. Add 0 to sum: 4

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

$1 \times 32 = 32$. Add 32 to sum: 36

$0 \times 64 = 0$. Add 0 to sum: 36

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

- 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

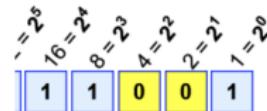
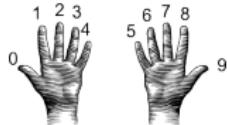
$0 \times 16 = 0$. Add 0 to sum: 4

$1 \times 32 = 32$. Add 32 to sum: 36

$0 \times 64 = 0$. Add 0 to sum: 36

$1 \times 128 = 0$. Add 128 to sum:

Binary to Decimal: Converting Between Bases

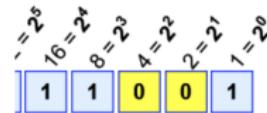
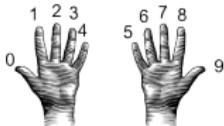


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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
$1 \times 128 = 0.$ Add 128 to sum:	164

Binary to Decimal: Converting Between Bases



Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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.

Recap

- Searching through data is a common task— built-in functions and standard design patterns for this.



Recap



- Searching through data is a common task— built-in functions and standard design patterns for this.
- Programming languages can be classified by the level of abstraction and direct access to data.

Recap



- Searching through data is a common task— built-in functions and standard design patterns for this.
- Programming languages can be classified by the level of abstraction and direct access to data.
- WeMIPS simplified machine language

Recap



- Searching through data is a common task— built-in functions and standard design patterns for this.
- Programming languages can be classified by the level of abstraction and direct access to data.
- WeMIPS simplified machine language
- Converting between Bases

Final Overview: Format

- The exam is 2 hours long.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.

Final Overview: Format

- The exam is 2 hours long.
 - There are 4 different versions to discourage copying.
 - It is on paper. No use of computers, phones, etc. allowed.
 - You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
 - ▶ More on logistics next lecture.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ No origami— it's distracting to others taking the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
 - ▶ More on logistics next lecture.
- Past exams available on webpage (includes answer keys).

Exam Options

Exam Times:

FINAL EXAM, VERSION 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper folded in half.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until I instruct you to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication during examinations) and fabrication of records and official documents as serious violations of academic integrity. The College's policy on academic dishonesty can be found in the CUNY Policy on Academic Honesty and will pursue cases of academic dishonesty according to the Hunter College

I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.	
Name:	
Sophia	
Email:	
Signature:	

Exam Options

Exam Times:

- Default Regular Time: Monday, 20 December, 9-11am.

FINAL EXAM, VERSION 3
CSci 122: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that you can write on.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication during examinations) and fabrication of records and official documents as serious violations of academic integrity. The College's policy on academic dishonesty is set forth in the CUNY Policy on Academic Honesty and will pursue cases of academic dishonesty according to the Hunter College

I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.	
Name:	
Sophomore:	
Email:	
Signature:	

Exam Options

Exam Times:

- Default Regular Time: Monday, 20 December, 9-11am.
- Alternate Time: Friday, 17 December, 8am-10am.

FINAL EXAM, VERSION 3
CSci 122: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that you can write on.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart phone, or other electronic device.
- Do not open this exam until I instruct you to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication during examinations) and fabrication of records and official documents as serious violations of academic integrity. The College's policy on academic dishonesty can be found in the CUNY Policy on Academic Honesty and will pursue cases of academic dishonesty according to the Hunter College

I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.	
Name:	
Sophia	
Email:	
Signature:	

Exam Options

Exam Times:

- Default Regular Time: Monday, 20 December, 9-11am.
- Alternate Time: Friday, 17 December, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 30 November. If you have not done so already, email me no later than 23 November.

FINAL EXAM, VERSION 3
CSci 122: Introduction to Computer Science
Hunter College, City University of New York

19 December 2021

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper folded in half.
- When taking the exam, you may bring with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until I instruct you to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication among students during examinations and assignments) and official documents as serious violations of academic integrity. The College has a zero tolerance for CUEP Policy on Academic Honesty and will pursue cases of academic dishonesty according to the Hunter College

I acknowledge that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.	
Name:	
Sophia	
Email:	
Signature:	

Exam Options

Exam Times:

- Default Regular Time: Monday, 20 December, 9-11am.
- Alternate Time: Friday, 17 December, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 30 November. If you have not done so already, email me no later than 23 November.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on 20 December.**

FINAL EXAM, VERSION 3
CSci 122: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper folded in half.
- When taking the exam, you may have only your pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart phone, or other electronic device.
- Do not open this exam until I instruct you to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication during examinations) and fabrication of records and official documents as serious violations of academic integrity. The College has adopted the CUNY Policy on Academic Honesty and will pursue cases of academic dishonesty according to the Hunter College

Indicate that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name: _____
Signature: _____
_____ _____ _____ _____ _____ _____

Exam Options

Exam Times:

- Default Regular Time: Monday, 20 December, 9-11am.
- Alternate Time: Friday, 17 December, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 30 November. If you have not done so already, email me no later than 23 November.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on 20 December.**
- If you choose to take the early date, **you will not be given access to the exam on 20 December even if you miss the early exam.**

FINAL EXAM, VERSION 3
CSci 122: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that you can write on.
- When taking the exam, you may have your pen and pencil, and your note sheet.
- You may not use a computer, calculator, tablet, smart phone, or other electronic device.
- Do not open this exam until I instruct you to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication during examinations) and fabrication of records and official documents as serious violations of academic integrity. The College has a zero tolerance policy towards acts of academic dishonesty and will pursue cases of academic dishonesty according to the Hunter College

Indicate that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name: _____
Signature: _____
_____ _____ _____ _____ _____ _____

Exam Options

Exam Times:

- Default Regular Time: Monday, 20 December, 9-11am.
- Alternate Time: Friday, 17 December, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 30 November. If you have not done so already, email me no later than 23 November.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on 20 December.**
- If you choose to take the early date, **you will not be given access to the exam on 20 December even if you miss the early exam.**

FINAL EXAM, VERSION 3
CSci 122: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that you can write on.
- When taking the exam, you may have your pen and pencil, and your note sheet.
- You may not use a computer, calculator, tablet, smart phone, or other electronic device.
- Do not open this exam until I instruct you to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communication during examinations) and fabrication of records and official documents as serious violations of academic integrity. The College has a zero tolerance policy towards acts of academic dishonesty and will pursue cases of academic dishonesty according to the Hunter College

Indicate that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name: _____
Signature: _____
_____ _____ _____ _____ _____ _____

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (one every two weeks) in lab 1001E Hunter North

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (one every two weeks) in lab 1001E Hunter North
- Submit this week's 5 programming assignments (programs 51-55)

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every two weeks**) in lab 1001E Hunter North
- Submit this week's 5 programming assignments (programs 51-55)
- If you need help, schedule an appointment for Tutoring in lab 1001E 11am-5pm

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every two weeks**) in lab 1001E Hunter North
- Submit this week's 5 programming assignments (programs 51-55)
- If you need help, schedule an appointment for Tutoring in lab 1001E 11am-5pm
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)