

# CSci 127: Introduction to Computer Science



[hunter.cuny.edu/csci](http://hunter.cuny.edu/csci)

# 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, May 22 from 9-11 am.*

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, May 22 from 9-11 am.*

*The early final exam (alternative date) is on Wednesday, May 17 in 1001G (time TBD).*

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, May 22 from 9-11 am.*

*The early final exam (alternative date) is on Wednesday, May 17 in 1001G (time TBD).*

*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, May 22 from 9-11 am.*

*The early final exam (alternative date) is on Wednesday, May 17 in 1001G (time TBD).*

*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, May 22 from 9-11 am.*

*The early final exam (alternative date) is on Wednesday, May 17 in 1001G (time TBD).*

*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, May 22 from 9-11 am.*

*The early final exam (alternative date) is on Wednesday, May 17 in 1001G (time TBD).*

*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.*
- ▶ *To help practice, there will be a mock exam during our last meeting on May 16.*

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, May 22 from 9-11 am.*

*The early final exam (alternative date) is on Wednesday, May 17 in 1001G (time TBD).*

*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.*
- ▶ *To help practice, there will be a mock exam during our last meeting on May 16.*
- ▶ *The mock exam will be run exactly like the real final.*

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, May 22 from 9-11 am.*

*The early final exam (alternative date) is on Wednesday, May 17 in 1001G (time TBD).*

*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.
- ▶ To help practice, there will be a mock exam during our last meeting on May 16.
- ▶ The mock exam will be run exactly like the real final.
- ▶ If you are already acquainted with the logistics you will have less stress during the real event.

# 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 Python code editor interface. On the left, the code file 'main.py' is open, containing 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)
```

On the right, the 'Result' tab displays the output of the script: a purple regular hexagon drawn on the screen, with each vertex marked by a 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.

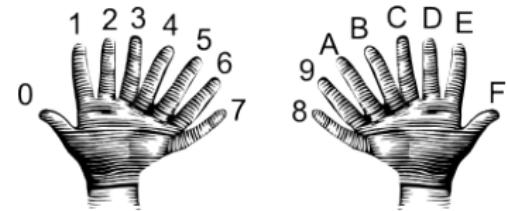
## Week 2: variables, data types, more on loops & range()

- 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)
```

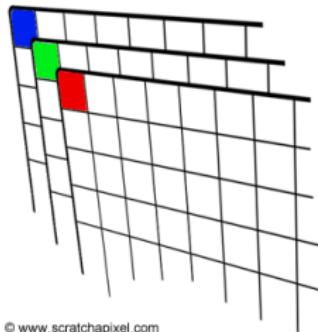
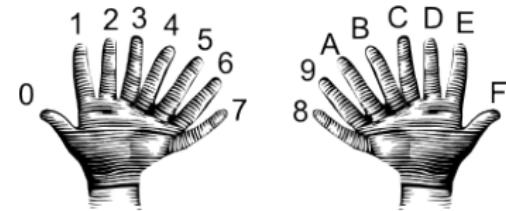
# Week 3: colors, hex, slices, numpy & images

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



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

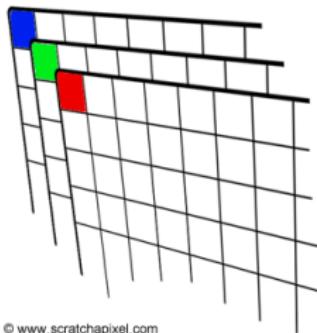
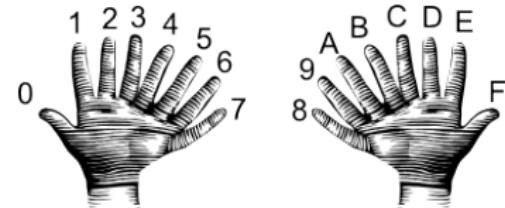
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



© www.scratchapixel.com

# 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.")

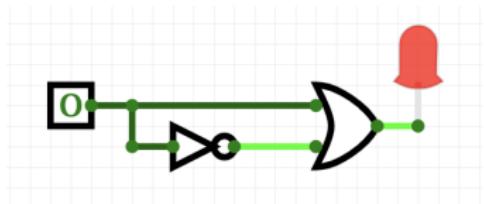
visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

# 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](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  
1690,4937,2037,727,788,1000  
1771,21843,3623,1,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67500,6250,6840,2000,49734  
1820,123704,11487,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14000,5348,10965,391114  
1850,355441,21800,18500,5800,11500,50015  
1860,513469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1367711,70000,65000,58000,41000,2150004  
1900,185093,116582,152999,200567,67621,2437202  
1910,223342,1634351,284041,430980,8569,4766803  
1920,2210103,2018354,446000,720201,116000,500000  
1930,1867128,1797128,235254,158200,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7891957  
1960,1690000,23195,1809000,1400000,1200000,781984  
1970,1539231,2460700,1471071,135443,768460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419782,8080879  
2010,1583873,2504705,2216722,1385108,461751,8175133  
2015,1444518,2436733,2339150,1454446,474558,8059405

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](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,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1890,4937,2037,727,788,1000  
1891,21843,3623,1,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,63545,5540,6242,1755,4543,75934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14049,5346,10965,391114  
1850,35549,12891,12891,12891,12891,12891  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911803  
1890,1367711,66582,116582,116582,116582,116582  
1900,185093,116582,152999,200567,67621,2437202  
1910,223342,1634351,2841,430980,8569,4766803  
1920,2210103,2018354,44607,72021,116582,50048  
1930,1867132,1797128,1797128,1797128,1797128,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1696131,2000239,1809001,1460711,2000239,781984  
1970,1539231,1865071,1747101,1472701,135443,768460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,379977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2277722,1385108,4175133,8175133  
2015,1444018,2436733,2339150,1459444,474558,8056405

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  
1690,4937,2037,727,788,111  
1771,21843,3623,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67541,6211,6800,1755,4543,79334  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,19013,14087,5346,10965,391114  
1850,355441,21890,18897,5346,10965,415115  
1860,613469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911803  
1890,1340000,700000,600000,51980,33029,2151134  
1900,1850593,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44607,44607,73201,11651,50048  
1930,18671128,1796128,1796128,1796128,1796128,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690231,2093175,1890949,1690231,1890949,781984  
1970,1539231,2465705,2147205,1472705,135443,7071846  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,379877,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,4175133,8175133  
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,Population  
1690,203,2037,...,727,7181  
1771,21843,28232,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,72000,6350,7000,1800,5000,89734  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,142278  
1840,311510,11013,14000,5348,10965,391114  
1850,35549,128000,128000,128000,128000,128000  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33051,1911801  
1890,1370000,660000,660000,660000,660000,660000  
1900,1850093,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44600,44600,73201,11651,50048  
1930,26671128,2079128,2079128,2079128,2079128,5930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690000,2319319,1890000,1890000,1890000,1890000,1890000  
1970,1539231,2465701,2465701,2465701,2465701,2465443,798460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,737997,77232564  
2000,1537195,2485326,2223979,1332650,419728,8080879  
2010,1583873,2504705,2277272,1385108,474558,8175133  
2015,1444018,2646733,2339150,1459446,474558,8056405
```

nycHistPop.csv

In Lab 6

# Week 6: structured data, pandas, & more design

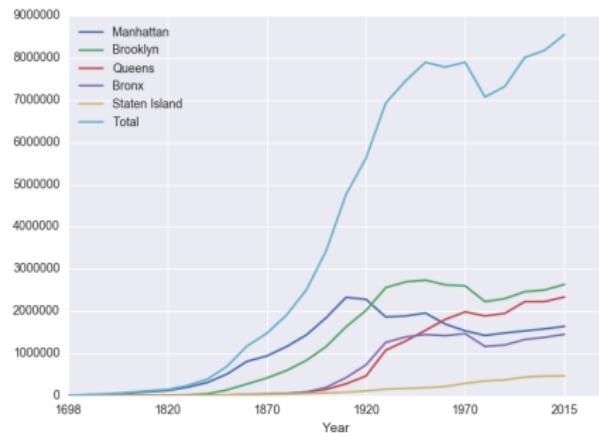
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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,Borough,Population  
1698,Manhattan,2037,727,7181  
1771,21843,36232,,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70515,6340,7042,1755,4543,75934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,12013,14034,5346,10965,391114  
1850,355449,128000,14895,5834,11545,411154  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1384473,72000,6353,5653,51980,33091,1911814  
1900,1850993,1165852,152999,200567,67921,2437202  
1910,233142,1634351,2841,430980,8569,476683  
1920,2210103,2018354,446071,446071,732013,1165852,152999  
1930,2667103,2486712,570128,570128,58345,58345,6930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7091957  
1960,1696101,2738175,1550849,1451277,191555,7091957  
1970,1696101,2738175,1550849,1451277,191555,7091984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2272722,1385108,451558,8175133  
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)
print('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



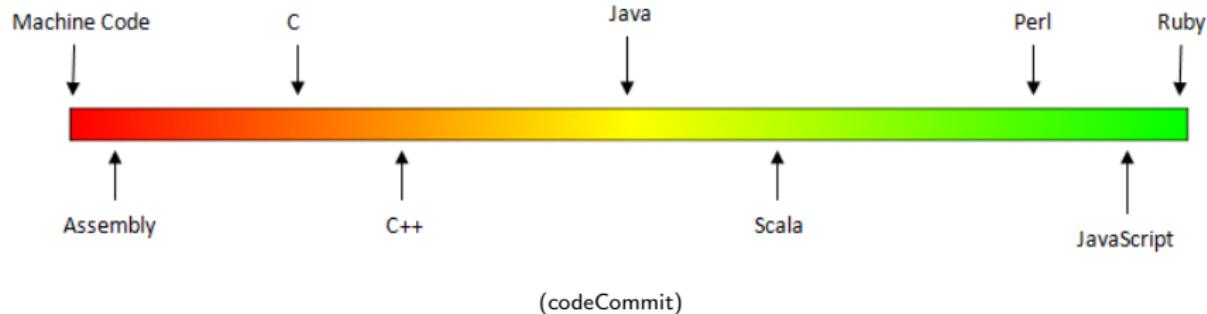
- Input/Output (I/O): `input()` and `print()`; pandas for CSV files
- Types:
  - ▶ Primitive: `int`, `float`, `bool`, `string`;
  - ▶ Container: lists (but not dictionaries/hashes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`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

# Today's Topics



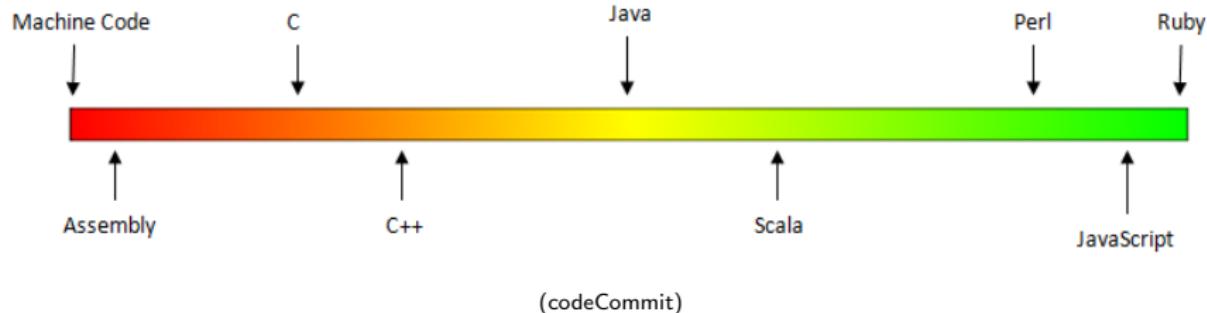
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic

# Low-Level vs. High-Level Languages



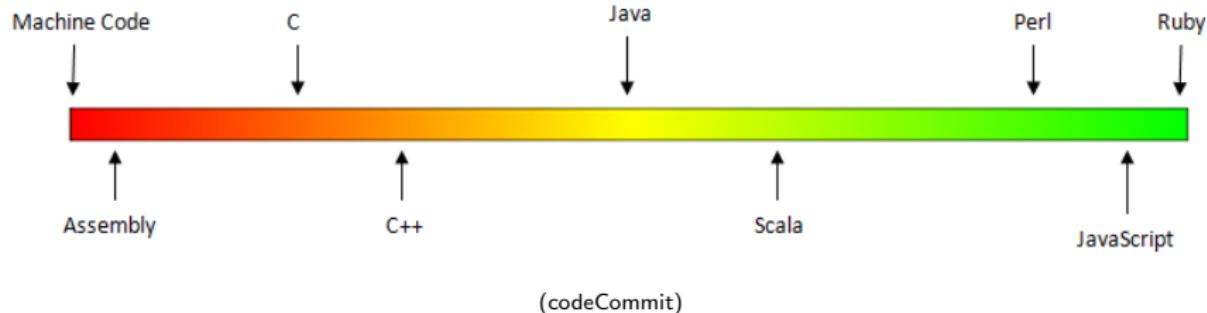
- Can view programming languages on a continuum.

# Low-Level vs. High-Level Languages



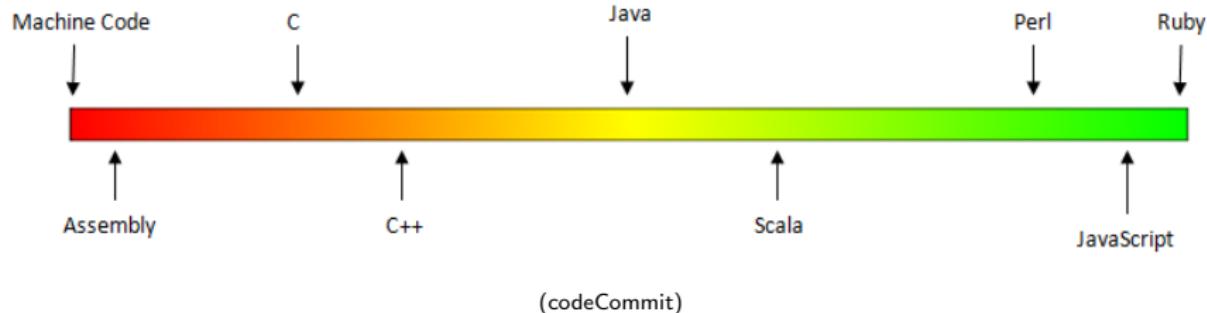
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

# Low-Level vs. High-Level Languages



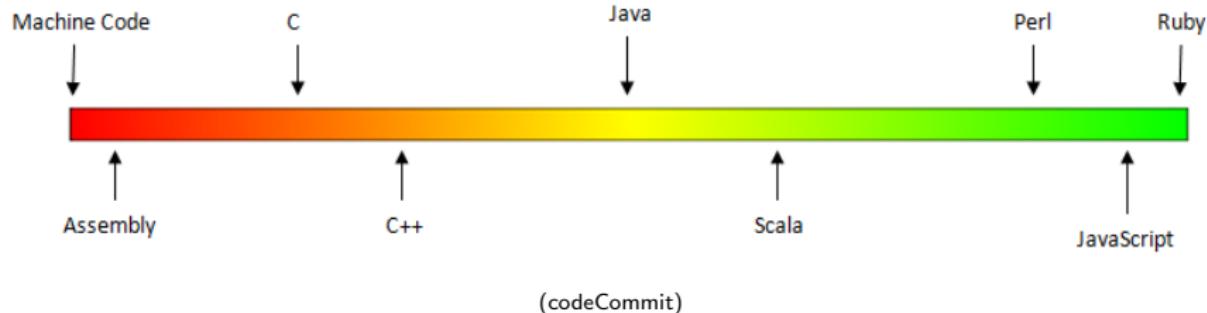
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

# 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.

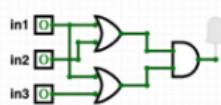
# Processing

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

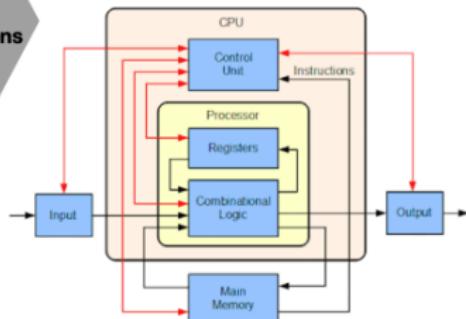


Data  
&  
Instructions

```
11 10 100 1110 10011000010  
11 00 10001101101110110111  
00 111010001101000 101011  
0010111010 00010000111111  
11101101111010001111011111  
0010110110 1011101 1001100  
0110011011011010010000001  
11110 1000000 101 10011101  
01100 0110110110001110101  
0100001000000011000100 000  
0110010101 10011 11001111  
10010 01000000 101110011101  
011011100 0101000 10000001  
1110001000000000000011100  
011000110 101111011011101  
0000011101001011001010 1110  
011011 01100000110110  
100 01 000101100101 001000  
01100110 0100000101 1001  
100101100 0110111010110110 1  
01101100100 01011001001  
11100110 0110110111011011  
01111010001 00000100001101  
0010 01000000100 1001101  
00010 1 10010011 100100100  
01110100011011110111010000  
1 11010101110011101 001101  
0111010010101 100 100100101  
000000110110001 10110 001
```

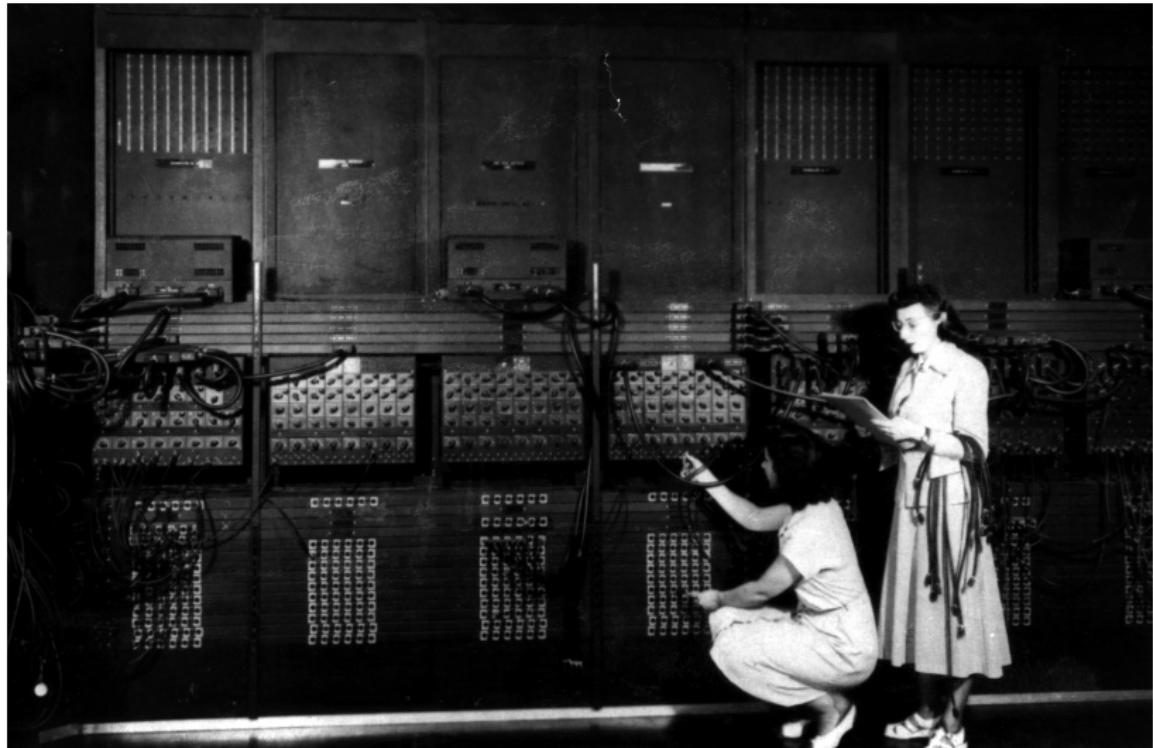


**Circuits (switches)**  
**On/Off 1/0 Logic**  
**Billions of switches/bits**



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

# Machine Language



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

# Machine Language

```
I FOX 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 00:UU .....
```

(wiki)

# Machine Language

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

```

A 002000 C2 3B REP #3B
A 002002 SED
A 002003 CLC
A 002004 FB
A 002005 34 12 ADD #1234
A 002007 69 21 43 ADC #43
A 002008 69 16 7F STA #7F
A 00200E D9 CLD
A 00200F E2 3B SEP #3B
A 002011 90 BHK
A 002012 3BZ

```

F

```

PB PC Mm0:012C A X Y SP DP IB
; 00 0012 00100000 0000 0000 0002 CFFF 0000 00
$ 2800

BREAK

PB PC Mm0:012C A X Y SP DP IB
; 00 2013 00100000 5555 0000 0002 CFFF 0000 00
$ 7793 7793

Mm0:012C 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.

```

; 00 002000 C2 3B REP #43B
; 00 002002 1B CLC
; 00 002003 FB SED
; 00 002004 34 42 LM #01234
; 00 002005 69 21 01 ADD #1
; 00 002006 69 B1 77 01 STA #0177F93
; 00 002007 D9 CLD
; 00 002008 E2 3B SUB #3B
; 00 002009 90 00 BPK
; 00 002011 90 00 BPK
; 00 002012 90 00 BPK

F PB PC MMw=012C .A X Y SP DP IB
; 00 002012 00110000 0000 0000 0002 CF7F 0000 00
; 00 002000 7793

BREAK

PB PC MMw=012C .A X Y SP DP IB
; 00 2013 00110000 5555 0000 0002 CF7F 0000 00
; 00 002000 7793
; 00 002000 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

(wiki)

# Machine Language



(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

(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	<input checked="" type="checkbox"/> 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

```
User Guide | Unit Tests | Index
```

Line 3	Qut	Show/Hide Demo	User Guide   Unit Tests   <a href="#">Index</a>					
		Addition Doubler	Star Looper	Stack Test	Hello World			
		Code Gen Save String	Interactive	Binary Decimal	Decimal Binary			
		Debug						
# Store 'Hello world!' at the top of the stack								
ADDI \$t0, \$zero, 72 # \$t0								
LD \$t0, 0(\$sp)								
ADD \$t0, \$t0, 151 # \$t0								
SW \$t0, 1(\$sp)								
ADD \$t0, \$t0, 108 # \$t0								
SW \$t0, 2(\$sp)								
ADD \$t0, \$t0, 109 # \$t0								
SW \$t0, 3(\$sp)								
ADD \$t0, \$t0, 111 # \$t0								
SW \$t0, 4(\$sp)								
ADD \$t0, \$t0, 32 # \$(spare)								
SW \$t0, 5(\$sp)								
ADD \$t0, \$t0, 119 # \$t0								
SW \$t0, 6(\$sp)								
ADD \$t0, \$t0, 110 # \$t0								
SW \$t0, 7(\$sp)								
ADD \$t0, \$t0, 114 # \$t0								
SW \$t0, 8(\$sp)								
ADD \$t0, \$t0, 108 # \$t0								
SW \$t0, 9(\$sp)								
ADD \$t0, \$t0, 109 # \$t0								
SW \$t0, 10(\$sp)								
ADD \$t0, \$t0, 33 # \$t0								
SW \$t0, 11(\$sp)								
ADD \$t0, \$t0, 0 # \$(null)								
SW \$t0, 12(\$sp)								
ADDI \$t0, \$zero, 4 # 4 is for print string								
SW \$t0, 0(\$sp)								
JAL \$t0, 0(\$sp) # point to the log								
RECALL								
Step Run <input checked="" type="checkbox"/> Enable auto switching								
S	T	A	V	Stack	Log			
\$t0	10							
\$t1	0							
\$t2	0							
\$t3	22							
\$t4	000							
\$t5	976							
\$t6	977							
\$t7	418							

## (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', and tabs for 'Addition Counter', 'Btav', 'Looper', 'Stack Test', 'Hello World', 'Code Gen Save String', 'Interactive', 'Binary Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with icons for 'Run', 'Break', 'Step', 'Stop', and 'Reset'. To the right of the toolbar are links for 'User Guide', 'Unit Tests', and 'Docs'.

The main area contains assembly code:

```
1 # Shows 'Hello world!' at the top of the stack
2 .text
3 .globl _start
4 _start:
5    addi   $t0, $zero, 100 # $t0 = 100
6    addi   $t1, $zero, 101 # $t1 = 101
7    addi   $t2, $zero, 102 # $t2 = 102
8    addi   $t3, $zero, 103 # $t3 = 103
9    addi   $t4, $zero, 104 # $t4 = 104
10   addi   $t5, $zero, 105 # $t5 = 105
11   addi   $t6, $zero, 106 # $t6 = 106
12   addi   $t7, $zero, 107 # $t7 = 107
13   addi   $t8, $zero, 108 # $t8 = 108
14   addi   $t9, $zero, 109 # $t9 = 109
15   addi   $t10, $zero, 110 # $t10 = 110
16   addi   $t11, $zero, 111 # $t11 = 111
17   addi   $t12, $zero, 112 # $t12 = 112
18   addi   $t13, $zero, 113 # $t13 = 113
19   addi   $t14, $zero, 114 # $t14 = 114
20   addi   $t15, $zero, 115 # $t15 = 115
21   addi   $t16, $zero, 116 # $t16 = 116
22   addi   $t17, $zero, 117 # $t17 = 117
23   addi   $t18, $zero, 118 # $t18 = 118
24   addi   $t19, $zero, 119 # $t19 = 119
25   addi   $t20, $zero, 120 # $t20 = 120
26   addi   $t21, $zero, 121 # $t21 = 121
27   addi   $t22, $zero, 122 # $t22 = 122
28   addi   $t23, $zero, 123 # $t23 = 123
29   addi   $t24, $zero, 124 # $t24 = 124
30   addi   $t25, $zero, 125 # $t25 = 125
31   addi   $t26, $zero, 126 # $t26 = 126
32   addi   $t27, $zero, 127 # $t27 = 127
33   addi   $t28, $zero, 128 # $t28 = 128
34   addi   $t29, $zero, 129 # $t29 = 129
35   addi   $t30, $zero, 130 # $t30 = 130
36   addi   $t31, $zero, 131 # $t31 = 131
37   addi   $t32, $zero, 132 # $t32 = 132
38   addi   $t33, $zero, 133 # $t33 = 133
39   addi   $t34, $zero, 134 # $t34 = 134
40   addi   $t35, $zero, 135 # $t35 = 135
41   addi   $t36, $zero, 136 # $t36 = 136
42   addi   $t37, $zero, 137 # $t37 = 137
43   addi   $t38, $zero, 138 # $t38 = 138
44   addi   $t39, $zero, 139 # $t39 = 139
45   addi   $t40, $zero, 140 # $t40 = 140
46   addi   $t41, $zero, 141 # $t41 = 141
47   addi   $t42, $zero, 142 # $t42 = 142
48   addi   $t43, $zero, 143 # $t43 = 143
49   addi   $t44, $zero, 144 # $t44 = 144
50   addi   $t45, $zero, 145 # $t45 = 145
51   addi   $t46, $zero, 146 # $t46 = 146
52   addi   $t47, $zero, 147 # $t47 = 147
53   addi   $t48, $zero, 148 # $t48 = 148
54   addi   $t49, $zero, 149 # $t49 = 149
55   addi   $t50, $zero, 150 # $t50 = 150
56   addi   $t51, $zero, 151 # $t51 = 151
57   addi   $t52, $zero, 152 # $t52 = 152
58   addi   $t53, $zero, 153 # $t53 = 153
59   addi   $t54, $zero, 154 # $t54 = 154
60   addi   $t55, $zero, 155 # $t55 = 155
61   addi   $t56, $zero, 156 # $t56 = 156
62   addi   $t57, $zero, 157 # $t57 = 157
63   addi   $t58, $zero, 158 # $t58 = 158
64   addi   $t59, $zero, 159 # $t59 = 159
65   addi   $t60, $zero, 160 # $t60 = 160
66   addi   $t61, $zero, 161 # $t61 = 161
67   addi   $t62, $zero, 162 # $t62 = 162
68   addi   $t63, $zero, 163 # $t63 = 163
69   addi   $t64, $zero, 164 # $t64 = 164
70   addi   $t65, $zero, 165 # $t65 = 165
71   addi   $t66, $zero, 166 # $t66 = 166
72   addi   $t67, $zero, 167 # $t67 = 167
73   addi   $t68, $zero, 168 # $t68 = 168
74   addi   $t69, $zero, 169 # $t69 = 169
75   addi   $t70, $zero, 170 # $t70 = 170
76   addi   $t71, $zero, 171 # $t71 = 171
77   addi   $t72, $zero, 172 # $t72 = 172
78   addi   $t73, $zero, 173 # $t73 = 173
79   addi   $t74, $zero, 174 # $t74 = 174
80   addi   $t75, $zero, 175 # $t75 = 175
81   addi   $t76, $zero, 176 # $t76 = 176
82   addi   $t77, $zero, 177 # $t77 = 177
83   addi   $t78, $zero, 178 # $t78 = 178
84   addi   $t79, $zero, 179 # $t79 = 179
85   addi   $t80, $zero, 180 # $t80 = 180
86   addi   $t81, $zero, 181 # $t81 = 181
87   addi   $t82, $zero, 182 # $t82 = 182
88   addi   $t83, $zero, 183 # $t83 = 183
89   addi   $t84, $zero, 184 # $t84 = 184
90   addi   $t85, $zero, 185 # $t85 = 185
91   addi   $t86, $zero, 186 # $t86 = 186
92   addi   $t87, $zero, 187 # $t87 = 187
93   addi   $t88, $zero, 188 # $t88 = 188
94   addi   $t89, $zero, 189 # $t89 = 189
95   addi   $t90, $zero, 190 # $t90 = 190
96   addi   $t91, $zero, 191 # $t91 = 191
97   addi   $t92, $zero, 192 # $t92 = 192
98   addi   $t93, $zero, 193 # $t93 = 193
99   addi   $t94, $zero, 194 # $t94 = 194
100  addi   $t95, $zero, 195 # $t95 = 195
101  addi   $t96, $zero, 196 # $t96 = 196
102  addi   $t97, $zero, 197 # $t97 = 197
103  addi   $t98, $zero, 198 # $t98 = 198
104  addi   $t99, $zero, 199 # $t99 = 199
105  addi   $t100, $zero, 200 # $t100 = 200
106  addi   $t101, $zero, 201 # $t101 = 201
107  addi   $t102, $zero, 202 # $t102 = 202
108  addi   $t103, $zero, 203 # $t103 = 203
109  addi   $t104, $zero, 204 # $t104 = 204
110  addi   $t105, $zero, 205 # $t105 = 205
111  addi   $t106, $zero, 206 # $t106 = 206
112  addi   $t107, $zero, 207 # $t107 = 207
113  addi   $t108, $zero, 208 # $t108 = 208
114  addi   $t109, $zero, 209 # $t109 = 209
115  addi   $t110, $zero, 210 # $t110 = 210
116  addi   $t111, $zero, 211 # $t111 = 211
117  addi   $t112, $zero, 212 # $t112 = 212
118  addi   $t113, $zero, 213 # $t113 = 213
119  addi   $t114, $zero, 214 # $t114 = 214
120  addi   $t115, $zero, 215 # $t115 = 215
121  addi   $t116, $zero, 216 # $t116 = 216
122  addi   $t117, $zero, 217 # $t117 = 217
123  addi   $t118, $zero, 218 # $t118 = 218
124  addi   $t119, $zero, 219 # $t119 = 219
125  addi   $t120, $zero, 220 # $t120 = 220
126  addi   $t121, $zero, 221 # $t121 = 221
127  addi   $t122, $zero, 222 # $t122 = 222
128  addi   $t123, $zero, 223 # $t123 = 223
129  addi   $t124, $zero, 224 # $t124 = 224
130  addi   $t125, $zero, 225 # $t125 = 225
131  addi   $t126, $zero, 226 # $t126 = 226
132  addi   $t127, $zero, 227 # $t127 = 227
133  addi   $t128, $zero, 228 # $t128 = 228
134  addi   $t129, $zero, 229 # $t129 = 229
135  addi   $t130, $zero, 230 # $t130 = 230
136  addi   $t131, $zero, 231 # $t131 = 231
137  addi   $t132, $zero, 232 # $t132 = 232
138  addi   $t133, $zero, 233 # $t133 = 233
139  addi   $t134, $zero, 234 # $t134 = 234
140  addi   $t135, $zero, 235 # $t135 = 235
141  addi   $t136, $zero, 236 # $t136 = 236
142  addi   $t137, $zero, 237 # $t137 = 237
143  addi   $t138, $zero, 238 # $t138 = 238
144  addi   $t139, $zero, 239 # $t139 = 239
145  addi   $t140, $zero, 240 # $t140 = 240
146  addi   $t141, $zero, 241 # $t141 = 241
147  addi   $t142, $zero, 242 # $t142 = 242
148  addi   $t143, $zero, 243 # $t143 = 243
149  addi   $t144, $zero, 244 # $t144 = 244
150  addi   $t145, $zero, 245 # $t145 = 245
151  addi   $t146, $zero, 246 # $t146 = 246
152  addi   $t147, $zero, 247 # $t147 = 247
153  addi   $t148, $zero, 248 # $t148 = 248
154  addi   $t149, $zero, 249 # $t149 = 249
155  addi   $t150, $zero, 250 # $t150 = 250
156  addi   $t151, $zero, 251 # $t151 = 251
157  addi   $t152, $zero, 252 # $t152 = 252
158  addi   $t153, $zero, 253 # $t153 = 253
159  addi   $t154, $zero, 254 # $t154 = 254
160  addi   $t155, $zero, 255 # $t155 = 255
161  addi   $t156, $zero, 256 # $t156 = 256
162  addi   $t157, $zero, 257 # $t157 = 257
163  addi   $t158, $zero, 258 # $t158 = 258
164  addi   $t159, $zero, 259 # $t159 = 259
165  addi   $t160, $zero, 260 # $t160 = 260
166  addi   $t161, $zero, 261 # $t161 = 261
167  addi   $t162, $zero, 262 # $t162 = 262
168  addi   $t163, $zero, 263 # $t163 = 263
169  addi   $t164, $zero, 264 # $t164 = 264
170  addi   $t165, $zero, 265 # $t165 = 265
171  addi   $t166, $zero, 266 # $t166 = 266
172  addi   $t167, $zero, 267 # $t167 = 267
173  addi   $t168, $zero, 268 # $t168 = 268
174  addi   $t169, $zero, 269 # $t169 = 269
175  addi   $t170, $zero, 270 # $t170 = 270
176  addi   $t171, $zero, 271 # $t171 = 271
177  addi   $t172, $zero, 272 # $t172 = 272
178  addi   $t173, $zero, 273 # $t173 = 273
179  addi   $t174, $zero, 274 # $t174 = 274
180  addi   $t175, $zero, 275 # $t175 = 275
181  addi   $t176, $zero, 276 # $t176 = 276
182  addi   $t177, $zero, 277 # $t177 = 277
183  addi   $t178, $zero, 278 # $t178 = 278
184  addi   $t179, $zero, 279 # $t179 = 279
185  addi   $t180, $zero, 280 # $t180 = 280
186  addi   $t181, $zero, 281 # $t181 = 281
187  addi   $t182, $zero, 282 # $t182 = 282
188  addi   $t183, $zero, 283 # $t183 = 283
189  addi   $t184, $zero, 284 # $t184 = 284
190  addi   $t185, $zero, 285 # $t185 = 285
191  addi   $t186, $zero, 286 # $t186 = 286
192  addi   $t187, $zero, 287 # $t187 = 287
193  addi   $t188, $zero, 288 # $t188 = 288
194  addi   $t189, $zero, 289 # $t189 = 289
195  addi   $t190, $zero, 290 # $t190 = 290
196  addi   $t191, $zero, 291 # $t191 = 291
197  addi   $t192, $zero, 292 # $t192 = 292
198  addi   $t193, $zero, 293 # $t193 = 293
199  addi   $t194, $zero, 294 # $t194 = 294
200  addi   $t195, $zero, 295 # $t195 = 295
201  addi   $t196, $zero, 296 # $t196 = 296
202  addi   $t197, $zero, 297 # $t197 = 297
203  addi   $t198, $zero, 298 # $t198 = 298
204  addi   $t199, $zero, 299 # $t199 = 299
205  addi   $t200, $zero, 300 # $t200 = 300
206  addi   $t201, $zero, 301 # $t201 = 301
207  addi   $t202, $zero, 302 # $t202 = 302
208  addi   $t203, $zero, 303 # $t203 = 303
209  addi   $t204, $zero, 304 # $t204 = 304
210  addi   $t205, $zero, 305 # $t205 = 305
211  addi   $t206, $zero, 306 # $t206 = 306
212  addi   $t207, $zero, 307 # $t207 = 307
213  addi   $t208, $zero, 308 # $t208 = 308
214  addi   $t209, $zero, 309 # $t209 = 309
215  addi   $t210, $zero, 310 # $t210 = 310
216  addi   $t211, $zero, 311 # $t211 = 311
217  addi   $t212, $zero, 312 # $t212 = 312
218  addi   $t213, $zero, 313 # $t213 = 313
219  addi   $t214, $zero, 314 # $t214 = 314
220  addi   $t215, $zero, 315 # $t215 = 315
221  addi   $t216, $zero, 316 # $t216 = 316
222  addi   $t217, $zero, 317 # $t217 = 317
223  addi   $t218, $zero, 318 # $t218 = 318
224  addi   $t219, $zero, 319 # $t219 = 319
225  addi   $t220, $zero, 320 # $t220 = 320
226  addi   $t221, $zero, 321 # $t221 = 321
227  addi   $t222, $zero, 322 # $t222 = 322
228  addi   $t223, $zero, 323 # $t223 = 323
229  addi   $t224, $zero, 324 # $t224 = 324
230  addi   $t225, $zero, 325 # $t225 = 325
231  addi   $t226, $zero, 326 # $t226 = 326
232  addi   $t227, $zero, 327 # $t227 = 327
233  addi   $t228, $zero, 328 # $t228 = 328
234  addi   $t229, $zero, 329 # $t229 = 329
235  addi   $t230, $zero, 330 # $t230 = 330
236  addi   $t231, $zero, 331 # $t231 = 331
237  addi   $t232, $zero, 332 # $t232 = 332
238  addi   $t233, $zero, 333 # $t233 = 333
239  addi   $t234, $zero, 334 # $t234 = 334
240  addi   $t235, $zero, 335 # $t235 = 335
241  addi   $t236, $zero, 336 # $t236 = 336
242  addi   $t237, $zero, 337 # $t237 = 337
243  addi   $t238, $zero, 338 # $t238 = 338
244  addi   $t239, $zero, 339 # $t239 = 339
245  addi   $t240, $zero, 340 # $t240 = 340
246  addi   $t241, $zero, 341 # $t241 = 341
247  addi   $t242, $zero, 342 # $t242 = 342
248  addi   $t243, $zero, 343 # $t243 = 343
249  addi   $t244, $zero, 344 # $t244 = 344
250  addi   $t245, $zero, 345 # $t245 = 345
251  addi   $t246, $zero, 346 # $t246 = 346
252  addi   $t247, $zero, 347 # $t247 = 347
253  addi   $t248, $zero, 348 # $t248 = 348
254  addi   $t249, $zero, 349 # $t249 = 349
255  addi   $t250, $zero, 350 # $t250 = 350
256  addi   $t251, $zero, 351 # $t251 = 351
257  addi   $t252, $zero, 352 # $t252 = 352
258  addi   $t253, $zero, 353 # $t253 = 353
259  addi   $t254, $zero, 354 # $t254 = 354
260  addi   $t255, $zero, 355 # $t255 = 355
261  addi   $t256, $zero, 356 # $t256 = 356
262  addi   $t257, $zero, 357 # $t257 = 357
263  addi   $t258, $zero, 358 # $t258 = 358
264  addi   $t259, $zero, 359 # $t259 = 359
265  addi   $t260, $zero, 360 # $t260 = 360
266  addi   $t261, $zero, 361 # $t261 = 361
267  addi   $t262, $zero, 362 # $t262 = 362
268  addi   $t263, $zero, 363 # $t263 = 363
269  addi   $t264, $zero, 364 # $t264 = 364
270  addi   $t265, $zero, 365 # $t265 = 365
271  addi   $t266, $zero, 366 # $t266 = 366
272  addi   $t267, $zero, 367 # $t267 = 367
273  addi   $t268, $zero, 368 # $t268 = 368
274  addi   $t269, $zero, 369 # $t269 = 369
275  addi   $t270, $zero, 370 # $t270 = 370
276  addi   $t271, $zero, 371 # $t271 = 371
277  addi   $t272, $zero, 372 # $t272 = 372
278  addi   $t273, $zero, 373 # $t273 = 373
279  addi   $t274, $zero, 374 # $t274 = 374
280  addi   $t275, $zero, 375 # $t275 = 375
281  addi   $t276, $zero, 376 # $t276 = 376
282  addi   $t277, $zero, 377 # $t277 = 377
283  addi   $t278, $zero, 378 # $t278 = 378
284  addi   $t279, $zero, 379 # $t279 = 379
285  addi   $t280, $zero, 380 # $t280 = 380
286  addi   $t281, $zero, 381 # $t281 = 381
287  addi   $t282, $zero, 382 # $t282 = 382
288  addi   $t283, $zero, 383 # $t283 = 383
289  addi   $t284, $zero, 384 # $t284 = 384
290  addi   $t285, $zero, 385 # $t285 = 385
291  addi   $t286, $zero, 386 # $t286 = 386
292  addi   $t287, $zero, 387 # $t287 = 387
293  addi   $t288, $zero, 388 # $t288 = 388
294  addi   $t289, $zero, 389 # $t289 = 389
295  addi   $t290, $zero, 390 # $t290 = 390
296  addi   $t291, $zero, 391 # $t291 = 391
297  addi   $t292, $zero, 392 # $t292 = 392
298  addi   $t293, $zero, 393 # $t293 = 393
299  addi   $t294, $zero, 394 # $t294 = 394
299  addi   $t295, $zero, 395 # $t295 = 395
299  addi   $t296, $zero, 396 # $t296 = 396
299  addi   $t297, $zero, 397 # $t297 = 397
299  addi   $t298, $zero, 398 # $t298 = 398
299  addi   $t299, $zero, 399 # $t299 = 399
299  addi   $t300, $zero, 400 # $t300 = 400
299  addi   $t301, $zero, 401 # $t301 = 401
299  addi   $t302, $zero, 402 # $t302 = 402
299  addi   $t303, $zero, 403 # $t303 = 403
299  addi   $t304, $zero, 404 # $t304 = 404
299  addi   $t305, $zero, 405 # $t305 = 405
299  addi   $t306, $zero, 406 # $t306 = 406
299  addi   $t307, $zero, 407 # $t307 = 407
299  addi   $t308, $zero, 408 # $t308 = 408
299  addi   $t309, $zero, 409 # $t309 = 409
299  addi   $t310, $zero, 410 # $t310 = 410
299  addi   $t311, $zero, 411 # $t311 = 411
299  addi   $t312, $zero, 412 # $t312 = 412
299  addi   $t313, $zero, 413 # $t313 = 413
299  addi   $t314, $zero, 414 # $t314 = 414
299  addi   $t315, $zero, 415 # $t315 = 415
299  addi   $t316, $zero, 416 # $t316 = 416
299  addi   $t317, $zero, 417 # $t317 = 417
299  addi   $t318, $zero, 418 # $t318 = 418
299  addi   $t319, $zero, 419 # $t319 = 419
299  addi   $t320, $zero, 420 # $t320 = 420
299  addi   $t321, $zero, 421 # $t321 = 421
299  addi   $t322, $zero, 422 # $t322 = 422
299  addi   $t323, $zero, 423 # $t323 = 423
299  addi   $t324, $zero, 424 # $t324 = 424
299  addi   $t325, $zero, 425 # $t325 = 425
299  addi   $t326, $zero, 426 # $t326 = 426
299  addi   $t327, $zero, 427 # $t327 = 427
299  addi   $t328, $zero, 428 # $t328 = 428
299  addi   $t329, $zero, 429 # $t329 = 429
299  addi   $t330, $zero, 430 # $t330 = 430
299  addi   $t331, $zero, 431 # $t331 = 431
299  addi   $t332, $zero, 432 # $t332 = 432
299  addi   $t333, $zero, 433 # $t333 = 433
299  addi   $t334, $zero, 434 # $t334 = 434
299  addi   $t335, $zero, 435 # $t335 = 435
299  addi   $t336, $zero, 436 # $t336 = 436
299  addi   $t337, $zero, 437 # $t337 = 437
299  addi   $t338, $zero, 438 # $t338 = 438
299  addi   $t339, $zero, 439 # $t339 = 439
299  addi   $t340, $zero, 440 # $t340 = 440
299  addi   $t341, $zero, 441 # $t341 = 441
299  addi   $t342, $zero, 442 # $t342 = 442
299  addi   $t343, $zero, 443 # $t343 = 443
299  addi   $t344, $zero, 444 # $t344 = 444
299  addi   $t345, $zero, 445 # $t345 = 445
299  addi   $t346, $zero, 446 # $t346 = 446
299  addi   $t347, $zero, 447 # $t347 = 447
299  addi   $t348, $zero, 448 # $t348 = 448
299  addi   $t349, $zero, 449 # $t349 = 449
299  addi   $t350, $zero, 450 # $t350 = 450
299  addi   $t351, $zero, 451 # $t351 = 451
299  addi   $t352, $zero, 452 # $t352 = 452
299  addi   $t353, $zero, 453 # $t353 = 453
299  addi   $t354, $zero, 454 # $t354 = 454
299  addi   $t355, $zero, 455 # $t355 = 455
299  addi   $t356, $zero, 456 # $t356 = 456
299  addi   $t357, $zero, 457 # $t357 = 457
299  addi   $t358, $zero, 458 # $t358 = 458
299  addi   $t359, $zero, 459 # $t359 = 459
299  addi   $t360, $zero, 460 # $t360 = 460
299  addi   $t361, $zero, 461 # $t361 = 461
299  addi   $t362, $zero, 462 # $t362 = 462
299  addi   $t363, $zero, 463 # $t363 = 463
299  addi   $t364, $zero, 464 # $t364 = 464
299  addi   $t365, $zero, 465 # $t365 = 465
299  addi   $t366, $zero, 466 # $t366 = 466
299  addi   $t367, $zero, 467 # $t367 = 467
299  addi   $t368, $zero, 468 # $t368 = 468
299  addi   $t369, $zero, 469 # $t369 = 469
299  addi   $t370, $zero, 470 # $t370 = 470
299  addi   $t371, $zero, 471 # $t371 = 471
299  addi   $t372, $zero, 472 # $t372 = 472
299  addi   $t373, $zero, 473 # $t373 = 473
299  addi   $t374, $zero, 474 # $t374 = 474
299  addi   $t375, $zero, 475 # $t375 = 475
299  addi   $t376, $zero, 476 # $t376 = 476
299  addi   $t377, $zero, 477 # $t377 = 477
299  addi   $t378, $zero, 478 # $t378 = 478
299  addi   $t379, $zero, 479 # $t379 = 479
299  addi   $t380, $zero, 480 # $t380 = 480
299  addi   $t381, $zero, 481 # $t381 = 481
299  addi   $t382, $zero, 482 # $t382 = 482
299  addi   $t383, $zero, 483 # $t383 = 483
299  addi   $t384, $zero, 484 # $t384 = 484
299  addi   $t385, $zero, 485 # $t385 = 485
299  addi   $t386, $zero, 486 # $t386 = 486
299  addi   $t387, $zero, 487 # $t387 = 487
299  addi   $t388, $zero, 488 # $t388 = 488
```

## MIPS Commands

- **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

- **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.

## 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

- **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.

# MIPS Commands

The screenshot shows the ShowMe Demo MIPS simulator. At the top, there are tabs for User, ShowMe Demo, Addition, Subtraction, Bitwise, Looper, Stack Test, Hello World, Code Gen, Save String, Interactive, Binary, Decimal, and Debug. Below the tabs is a menu bar with File, Run, Help, and Exit. On the right, there are links for User Guide and Unit Tests.

The assembly code window contains the following MIPS assembly code:

```
# Shows "Hello world" at the top of the stack
1.  .data
2.  msg: .asciiz "Hello world"
3.  .text
4.  addi $t0, $zero, 12 # $t0 = 12
5.  addi $t1, $zero, 13 # $t1 = 13
6.  addi $t2, $zero, 129 # $t2 = 129
7.  addi $t3, $zero, 128 # $t3 = 128
8.  addi $t4, $zero, 129 # $t4 = 129
9.  addi $t5, $zero, 128 # $t5 = 128
10. addi $t6, $zero, 129 # $t6 = 129
11. addi $t7, $zero, 128 # $t7 = 128
12. addi $t8, $zero, 129 # $t8 = 129
13. addi $t9, $zero, 128 # $t9 = 128
14. addi $t10, $zero, 129 # $t10 = 129
15. addi $t11, $zero, 128 # $t11 = 128
16. addi $t12, $zero, 129 # $t12 = 129
17. addi $t13, $zero, 128 # $t13 = 128
18. addi $t14, $zero, 129 # $t14 = 129
19. addi $t15, $zero, 128 # $t15 = 128
20. addi $t16, $zero, 129 # $t16 = 129
21. addi $t17, $zero, 128 # $t17 = 128
22. addi $t18, $zero, 129 # $t18 = 129
23. addi $t19, $zero, 128 # $t19 = 128
24. addi $t20, $zero, 129 # $t20 = 129
25. addi $t21, $zero, 128 # $t21 = 128
26. addi $t22, $zero, 129 # $t22 = 129
27. addi $t23, $zero, 128 # $t23 = 128
28. addi $t24, $zero, 129 # $t24 = 129
29. addi $t25, $zero, 128 # $t25 = 128
30. addi $t26, $zero, 129 # $t26 = 129
31. addi $t27, $zero, 128 # $t27 = 128
32. addi $t28, $zero, 129 # $t28 = 129
33. addi $t29, $zero, 128 # $t29 = 128
34. addi $t30, $zero, 129 # $t30 = 129
35. addi $t31, $zero, 128 # $t31 = 128
36. addi $t32, $zero, 129 # $t32 = 129
37. addi $t33, $zero, 128 # $t33 = 128
38. addi $t34, $zero, 129 # $t34 = 129
39. addi $t35, $zero, 128 # $t35 = 128
40. addi $t36, $zero, 129 # $t36 = 129
41. addi $t37, $zero, 128 # $t37 = 128
42. addi $t38, $zero, 129 # $t38 = 129
43. addi $t39, $zero, 128 # $t39 = 128
44. addi $t40, $zero, 129 # $t40 = 129
45. addi $t41, $zero, 128 # $t41 = 128
46. addi $t42, $zero, 129 # $t42 = 129
47. addi $t43, $zero, 128 # $t43 = 128
48. addi $t44, $zero, 129 # $t44 = 129
49. addi $t45, $zero, 128 # $t45 = 128
50. addi $t46, $zero, 129 # $t46 = 129
51. addi $t47, $zero, 128 # $t47 = 128
52. addi $t48, $zero, 129 # $t48 = 129
53. addi $t49, $zero, 128 # $t49 = 128
54. addi $t50, $zero, 129 # $t50 = 129
55. addi $t51, $zero, 128 # $t51 = 128
56. addi $t52, $zero, 129 # $t52 = 129
57. addi $t53, $zero, 128 # $t53 = 128
58. addi $t54, $zero, 129 # $t54 = 129
59. addi $t55, $zero, 128 # $t55 = 128
60. addi $t56, $zero, 129 # $t56 = 129
61. addi $t57, $zero, 128 # $t57 = 128
62. addi $t58, $zero, 129 # $t58 = 129
63. addi $t59, $zero, 128 # $t59 = 128
64. addi $t60, $zero, 129 # $t60 = 129
65. addi $t61, $zero, 128 # $t61 = 128
66. addi $t62, $zero, 129 # $t62 = 129
67. addi $t63, $zero, 128 # $t63 = 128
68. addi $t64, $zero, 129 # $t64 = 129
69. addi $t65, $zero, 128 # $t65 = 128
70. addi $t66, $zero, 129 # $t66 = 129
71. addi $t67, $zero, 128 # $t67 = 128
72. addi $t68, $zero, 129 # $t68 = 129
73. addi $t69, $zero, 128 # $t69 = 128
74. addi $t70, $zero, 129 # $t70 = 129
75. addi $t71, $zero, 128 # $t71 = 128
76. addi $t72, $zero, 129 # $t72 = 129
77. addi $t73, $zero, 128 # $t73 = 128
78. addi $t74, $zero, 129 # $t74 = 129
79. addi $t75, $zero, 128 # $t75 = 128
80. addi $t76, $zero, 129 # $t76 = 129
81. addi $t77, $zero, 128 # $t77 = 128
82. addi $t78, $zero, 129 # $t78 = 129
83. addi $t79, $zero, 128 # $t79 = 128
84. addi $t80, $zero, 129 # $t80 = 129
85. addi $t81, $zero, 128 # $t81 = 128
86. addi $t82, $zero, 129 # $t82 = 129
87. addi $t83, $zero, 128 # $t83 = 128
88. addi $t84, $zero, 129 # $t84 = 129
89. addi $t85, $zero, 128 # $t85 = 128
90. addi $t86, $zero, 129 # $t86 = 129
91. addi $t87, $zero, 128 # $t87 = 128
92. addi $t88, $zero, 129 # $t88 = 129
93. addi $t89, $zero, 128 # $t89 = 128
94. addi $t90, $zero, 129 # $t90 = 129
95. addi $t91, $zero, 128 # $t91 = 128
96. addi $t92, $zero, 129 # $t92 = 129
97. addi $t93, $zero, 128 # $t93 = 128
98. addi $t94, $zero, 129 # $t94 = 129
99. addi $t95, $zero, 128 # $t95 = 128
100. addi $t96, $zero, 129 # $t96 = 129
101. addi $t97, $zero, 128 # $t97 = 128
102. addi $t98, $zero, 129 # $t98 = 129
103. addi $t99, $zero, 128 # $t99 = 128
104. addi $t100, $zero, 129 # $t100 = 129
105. addi $t101, $zero, 128 # $t101 = 128
106. addi $t102, $zero, 129 # $t102 = 129
107. addi $t103, $zero, 128 # $t103 = 128
108. addi $t104, $zero, 129 # $t104 = 129
109. addi $t105, $zero, 128 # $t105 = 128
110. addi $t106, $zero, 129 # $t106 = 129
111. addi $t107, $zero, 128 # $t107 = 128
112. addi $t108, $zero, 129 # $t108 = 129
113. addi $t109, $zero, 128 # $t109 = 128
114. addi $t110, $zero, 129 # $t110 = 129
115. addi $t111, $zero, 128 # $t111 = 128
116. addi $t112, $zero, 129 # $t112 = 129
117. addi $t113, $zero, 128 # $t113 = 128
118. addi $t114, $zero, 129 # $t114 = 129
119. addi $t115, $zero, 128 # $t115 = 128
120. addi $t116, $zero, 129 # $t116 = 129
121. addi $t117, $zero, 128 # $t117 = 128
122. addi $t118, $zero, 129 # $t118 = 129
123. addi $t119, $zero, 128 # $t119 = 128
124. addi $t120, $zero, 129 # $t120 = 129
125. addi $t121, $zero, 128 # $t121 = 128
126. addi $t122, $zero, 129 # $t122 = 129
127. addi $t123, $zero, 128 # $t123 = 128
128. addi $t124, $zero, 129 # $t124 = 129
129. addi $t125, $zero, 128 # $t125 = 128
130. addi $t126, $zero, 129 # $t126 = 129
131. addi $t127, $zero, 128 # $t127 = 128
132. addi $t128, $zero, 129 # $t128 = 129
133. addi $t129, $zero, 128 # $t129 = 128
134. addi $t130, $zero, 129 # $t130 = 129
135. addi $t131, $zero, 128 # $t131 = 128
136. addi $t132, $zero, 129 # $t132 = 129
137. addi $t133, $zero, 128 # $t133 = 128
138. addi $t134, $zero, 129 # $t134 = 129
139. addi $t135, $zero, 128 # $t135 = 128
140. addi $t136, $zero, 129 # $t136 = 129
141. addi $t137, $zero, 128 # $t137 = 128
142. addi $t138, $zero, 129 # $t138 = 129
143. addi $t139, $zero, 128 # $t139 = 128
144. addi $t140, $zero, 129 # $t140 = 129
145. addi $t141, $zero, 128 # $t141 = 128
146. addi $t142, $zero, 129 # $t142 = 129
147. addi $t143, $zero, 128 # $t143 = 128
148. addi $t144, $zero, 129 # $t144 = 129
149. addi $t145, $zero, 128 # $t145 = 128
150. addi $t146, $zero, 129 # $t146 = 129
151. addi $t147, $zero, 128 # $t147 = 128
152. addi $t148, $zero, 129 # $t148 = 129
153. addi $t149, $zero, 128 # $t149 = 128
154. addi $t150, $zero, 129 # $t150 = 129
155. addi $t151, $zero, 128 # $t151 = 128
156. addi $t152, $zero, 129 # $t152 = 129
157. addi $t153, $zero, 128 # $t153 = 128
158. addi $t154, $zero, 129 # $t154 = 129
159. addi $t155, $zero, 128 # $t155 = 128
160. addi $t156, $zero, 129 # $t156 = 129
161. addi $t157, $zero, 128 # $t157 = 128
162. addi $t158, $zero, 129 # $t158 = 129
163. addi $t159, $zero, 128 # $t159 = 128
164. addi $t160, $zero, 129 # $t160 = 129
165. addi $t161, $zero, 128 # $t161 = 128
166. addi $t162, $zero, 129 # $t162 = 129
167. addi $t163, $zero, 128 # $t163 = 128
168. addi $t164, $zero, 129 # $t164 = 129
169. addi $t165, $zero, 128 # $t165 = 128
170. addi $t166, $zero, 129 # $t166 = 129
171. addi $t167, $zero, 128 # $t167 = 128
172. addi $t168, $zero, 129 # $t168 = 129
173. addi $t169, $zero, 128 # $t169 = 128
174. addi $t170, $zero, 129 # $t170 = 129
175. addi $t171, $zero, 128 # $t171 = 128
176. addi $t172, $zero, 129 # $t172 = 129
177. addi $t173, $zero, 128 # $t173 = 128
178. addi $t174, $zero, 129 # $t174 = 129
179. addi $t175, $zero, 128 # $t175 = 128
180. addi $t176, $zero, 129 # $t176 = 129
181. addi $t177, $zero, 128 # $t177 = 128
182. addi $t178, $zero, 129 # $t178 = 129
183. addi $t179, $zero, 128 # $t179 = 128
184. addi $t180, $zero, 129 # $t180 = 129
185. addi $t181, $zero, 128 # $t181 = 128
186. addi $t182, $zero, 129 # $t182 = 129
187. addi $t183, $zero, 128 # $t183 = 128
188. addi $t184, $zero, 129 # $t184 = 129
189. addi $t185, $zero, 128 # $t185 = 128
190. addi $t186, $zero, 129 # $t186 = 129
191. addi $t187, $zero, 128 # $t187 = 128
192. addi $t188, $zero, 129 # $t188 = 129
193. addi $t189, $zero, 128 # $t189 = 128
194. addi $t190, $zero, 129 # $t190 = 129
195. addi $t191, $zero, 128 # $t191 = 128
196. addi $t192, $zero, 129 # $t192 = 129
197. addi $t193, $zero, 128 # $t193 = 128
198. addi $t194, $zero, 129 # $t194 = 129
199. addi $t195, $zero, 128 # $t195 = 128
200. addi $t196, $zero, 129 # $t196 = 129
201. addi $t197, $zero, 128 # $t197 = 128
202. addi $t198, $zero, 129 # $t198 = 129
203. addi $t199, $zero, 128 # $t199 = 128
204. addi $t200, $zero, 129 # $t200 = 129
205. addi $t201, $zero, 128 # $t201 = 128
206. addi $t202, $zero, 129 # $t202 = 129
207. addi $t203, $zero, 128 # $t203 = 128
208. addi $t204, $zero, 129 # $t204 = 129
209. addi $t205, $zero, 128 # $t205 = 128
210. addi $t206, $zero, 129 # $t206 = 129
211. addi $t207, $zero, 128 # $t207 = 128
212. addi $t208, $zero, 129 # $t208 = 129
213. addi $t209, $zero, 128 # $t209 = 128
214. addi $t210, $zero, 129 # $t210 = 129
215. addi $t211, $zero, 128 # $t211 = 128
216. addi $t212, $zero, 129 # $t212 = 129
217. addi $t213, $zero, 128 # $t213 = 128
218. addi $t214, $zero, 129 # $t214 = 129
219. addi $t215, $zero, 128 # $t215 = 128
220. addi $t216, $zero, 129 # $t216 = 129
221. addi $t217, $zero, 128 # $t217 = 128
222. addi $t218, $zero, 129 # $t218 = 129
223. addi $t219, $zero, 128 # $t219 = 128
224. addi $t220, $zero, 129 # $t220 = 129
225. addi $t221, $zero, 128 # $t221 = 128
226. addi $t222, $zero, 129 # $t222 = 129
227. addi $t223, $zero, 128 # $t223 = 128
228. addi $t224, $zero, 129 # $t224 = 129
229. addi $t225, $zero, 128 # $t225 = 128
230. addi $t226, $zero, 129 # $t226 = 129
231. addi $t227, $zero, 128 # $t227 = 128
232. addi $t228, $zero, 129 # $t228 = 129
233. addi $t229, $zero, 128 # $t229 = 128
234. addi $t230, $zero, 129 # $t230 = 129
235. addi $t231, $zero, 128 # $t231 = 128
236. addi $t232, $zero, 129 # $t232 = 129
237. addi $t233, $zero, 128 # $t233 = 128
238. addi $t234, $zero, 129 # $t234 = 129
239. addi $t235, $zero, 128 # $t235 = 128
240. addi $t236, $zero, 129 # $t236 = 129
241. addi $t237, $zero, 128 # $t237 = 128
242. addi $t238, $zero, 129 # $t238 = 129
243. addi $t239, $zero, 128 # $t239 = 128
244. addi $t240, $zero, 129 # $t240 = 129
245. addi $t241, $zero, 128 # $t241 = 128
246. addi $t242, $zero, 129 # $t242 = 129
247. addi $t243, $zero, 128 # $t243 = 128
248. addi $t244, $zero, 129 # $t244 = 129
249. addi $t245, $zero, 128 # $t245 = 128
250. addi $t246, $zero, 129 # $t246 = 129
251. addi $t247, $zero, 128 # $t247 = 128
252. addi $t248, $zero, 129 # $t248 = 129
253. addi $t249, $zero, 128 # $t249 = 128
254. addi $t250, $zero, 129 # $t250 = 129
255. addi $t251, $zero, 128 # $t251 = 128
256. addi $t252, $zero, 129 # $t252 = 129
257. addi $t253, $zero, 128 # $t253 = 128
258. addi $t254, $zero, 129 # $t254 = 129
259. addi $t255, $zero, 128 # $t255 = 128
260. addi $t256, $zero, 129 # $t256 = 129
261. addi $t257, $zero, 128 # $t257 = 128
262. addi $t258, $zero, 129 # $t258 = 129
263. addi $t259, $zero, 128 # $t259 = 128
264. addi $t260, $zero, 129 # $t260 = 129
265. addi $t261, $zero, 128 # $t261 = 128
266. addi $t262, $zero, 129 # $t262 = 129
267. addi $t263, $zero, 128 # $t263 = 128
268. addi $t264, $zero, 129 # $t264 = 129
269. addi $t265, $zero, 128 # $t265 = 128
270. addi $t266, $zero, 129 # $t266 = 129
271. addi $t267, $zero, 128 # $t267 = 128
272. addi $t268, $zero, 129 # $t268 = 129
273. addi $t269, $zero, 128 # $t269 = 128
274. addi $t270, $zero, 129 # $t270 = 129
275. addi $t271, $zero, 128 # $t271 = 128
276. addi $t272, $zero, 129 # $t272 = 129
277. addi $t273, $zero, 128 # $t273 = 128
278. addi $t274, $zero, 129 # $t274 = 129
279. addi $t275, $zero, 128 # $t275 = 128
280. addi $t276, $zero, 129 # $t276 = 129
281. addi $t277, $zero, 128 # $t277 = 128
282. addi $t278, $zero, 129 # $t278 = 129
283. addi $t279, $zero, 128 # $t279 = 128
284. addi $t280, $zero, 129 # $t280 = 129
285. addi $t281, $zero, 128 # $t281 = 128
286. addi $t282, $zero, 129 # $t282 = 129
287. addi $t283, $zero, 128 # $t283 = 128
288. addi $t284, $zero, 129 # $t284 = 129
289. addi $t285, $zero, 128 # $t285 = 128
290. addi $t286, $zero, 129 # $t286 = 129
291. addi $t287, $zero, 128 # $t287 = 128
292. addi $t288, $zero, 129 # $t288 = 129
293. addi $t289, $zero, 128 # $t289 = 128
294. addi $t290, $zero, 129 # $t290 = 129
295. addi $t291, $zero, 128 # $t291 = 128
296. addi $t292, $zero, 129 # $t292 = 129
297. addi $t293, $zero, 128 # $t293 = 128
298. addi $t294, $zero, 129 # $t294 = 129
299. addi $t295, $zero, 128 # $t295 = 128
300. addi $t296, $zero, 129 # $t296 = 129
301. addi $t297, $zero, 128 # $t297 = 128
302. addi $t298, $zero, 129 # $t298 = 129
303. addi $t299, $zero, 128 # $t299 = 128
304. addi $t300, $zero, 129 # $t300 = 129
305. addi $t301, $zero, 128 # $t301 = 128
306. addi $t302, $zero, 129 # $t302 = 129
307. addi $t303, $zero, 128 # $t303 = 128
308. addi $t304, $zero, 129 # $t304 = 129
309. addi $t305, $zero, 128 # $t305 = 128
310. addi $t306, $zero, 129 # $t306 = 129
311. addi $t307, $zero, 128 # $t307 = 128
312. addi $t308, $zero, 129 # $t308 = 129
313. addi $t309, $zero, 128 # $t309 = 128
314. addi $t310, $zero, 129 # $t310 = 129
315. addi $t311, $zero, 128 # $t311 = 128
316. addi $t312, $zero, 129 # $t312 = 129
317. addi $t313, $zero, 128 # $t313 = 128
318. addi $t314, $zero, 129 # $t314 = 129
319. addi $t315, $zero, 128 # $t315 = 128
320. addi $t316, $zero, 129 # $t316 = 129
321. addi $t317, $zero, 128 # $t317 = 128
322. addi $t318, $zero, 129 # $t318 = 129
323. addi $t319, $zero, 128 # $t319 = 128
324. addi $t320, $zero, 129 # $t320 = 129
325. addi $t321, $zero, 128 # $t321 = 128
326. addi $t322, $zero, 129 # $t322 = 129
327. addi $t323, $zero, 128 # $t323 = 128
328. addi $t324, $zero, 129 # $t324 = 129
329. addi $t325, $zero, 128 # $t325 = 128
330. addi $t326, $zero, 129 # $t326 = 129
331. addi $t327, $zero, 128 # $t327 = 128
332. addi $t328, $zero, 129 # $t328 = 129
333. addi $t329, $zero, 128 # $t329 = 128
334. addi $t330, $zero, 129 # $t330 = 129
335. addi $t331, $zero, 128 # $t331 = 128
336. addi $t332, $zero, 129 # $t332 = 129
337. addi $t333, $zero, 128 # $t333 = 128
338. addi $t334, $zero, 129 # $t334 = 129
339. addi $t335, $zero, 128 # $t335 = 128
340. addi $t336, $zero, 129 # $t336 = 129
341. addi $t337, $zero, 128 # $t337 = 128
342. addi $t338, $zero, 129 # $t338 = 129
343. addi $t339, $zero, 128 # $t339 = 128
344. addi $t340, $zero, 129 # $t340 = 129
345. addi $t341, $zero, 128 # $t341 = 128
346. addi $t342, $zero, 129 # $t342 = 129
347. addi $t343, $zero, 128 # $t343 = 128
348. addi $t344, $zero, 129 # $t344 = 129
349. addi $t345, $zero, 128 # $t345 = 128
350. addi $t346, $zero, 129 # $t346 = 129
351. addi $t347, $zero, 128 # $t347 = 128
352. addi $t348, $zero, 129 # $t348 = 129
353. addi $t349, $zero, 128 # $t349 = 128
354. addi $t350, $zero, 129 # $t350 = 129
355. addi $t351, $zero, 128 # $t351 = 128
356. addi $t352, $zero, 129 # $t352 = 129
357. addi $t353, $zero, 128 # $t353 = 128
358. addi $t354, $zero, 129 # $t354 = 129
359. addi $t355, $zero, 128 # $t355 = 128
360. addi $t356, $zero, 129 # $t356 = 129
361. addi $t357, $zero, 128 # $t357 = 128
362. addi $t358, $zero, 129 # $t358 = 129
363. addi $t359, $zero, 128 # $t359 = 128
364. addi $t360, $zero, 129 # $t360 = 129
365. addi $t361, $zero, 128 # $t361 = 128
366. addi $t362, $zero, 129 # $t362 = 129
367. addi $t363, $zero, 128 # $t363 = 128
368. addi $t364, $zero, 129 # $t364 = 129
369. addi $t365, $zero, 128 # $t365 = 128
370. addi $t366, $zero, 129 # $t366 = 129
371. addi $t367, $zero, 128 # $t367 = 128
372. addi $t368, $zero, 129 # $t368 = 129
373. addi $t369, $zero, 128 # $t369 = 128
374. addi $t370, $zero, 129 # $t370 = 129
375. addi $t371, $zero, 128 # $t371 = 128
376. addi $t372, $zero, 129 # $t372 = 129
377. addi $t373, $zero, 128 # $t373 = 128
378. addi $t374, $zero, 129 # $t374 = 129
379. addi $t375, $zero, 128 # $t375 = 128
380. addi $t376, $zero, 129 # $t376 = 129
381. addi $t377, $zero, 128 # $t377 = 128
382. addi $t378, $zero, 129 # $t378 = 129
383. addi $t379, $zero, 128 # $t379 = 128
384. addi $t380, $zero, 129 # $t380 = 129
385. addi $t381, $zero, 128 # $t381 = 128
386. addi $t382, $zero, 129 # $t382 = 129
387. addi $t383, $zero, 128 # $t383 = 128
388. addi $t384, $zero, 129 # $t384 = 129
389. addi $t385, $zero, 128 # $t385 = 128
390. addi $t386, $zero, 129 # $t386 = 129
391. addi $t387, $zero, 128 # $t387 = 128
392. addi $t388, $zero, 129 # $t388 = 129
393. addi $t389, $zero, 128 # $t389 = 128
394. addi $t390, $zero, 129 # $t390 = 129
395. addi $t391, $zero, 128 # $t391 = 128
396. addi $t392, $zero, 129 # $t392 = 129
397. addi $t393, $zero, 128 # $t393 = 128
398. addi $t394, $zero, 129 # $t394 = 129
399. addi $t395, $zero, 128 # $t395 = 128
400. addi $t396, $zero, 129 # $t396 = 129
401. addi $t397, $zero, 128 # $t397 = 128
402. addi $t398, $zero, 129 # $t398 = 129
403. addi $t399, $zero, 128 # $t399 = 128
404. addi $t400, $zero, 129 # $t400 = 129
405. addi $t401, $zero, 128 # $t401 = 128
406. addi $t402, $zero,
```

## 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

```

Line 3 Out ShowWhile Demo User Guide | Unit Tests | Docs

Addition Doubler Stmt Looper Stack Test Hello World
Code Gen Save String Interactive Binary0 Decimal Decimal2 Binary

Debug Step Run | # Enable auto switching
S T A V Stack Log
#d 10
#e 8
#h 9
#b 22
#d 890
#b 976
#d 977
#f 419

# Prints "Hello world!" at the top of the stack
#EQU1 std::cout, 72 #'H
#EQU1 std::endl, 10 #'n
#EQU1 std::wcout, 100 #'\u000d
#EQU1 std::wendl, 101 #'\u000a
#EQU1 std::hex, 198 #'1
#EQU1 std::dec, 199 #'0
#EQU1 std::oct, 197 #'2
#EQU1 std::wcout, 111 #'o
#EQU1 std::wendl, 112 #'\u000a
#EQU1 std::wcout, 41 #'open parenthesis
#D 94s, 5(#expr)
#EQU1 std::wcout, 113 #'w
#EQU1 std::wcout, 114 #'w
#EQU1 std::wcout, 115 #'n
#EQU1 std::wcout, 71(#expr)
#EQU1 std::wcout, 116 #'c
#D 95s, 6(#expr)
#EQU1 std::wcout, 118 #'1
#EQU1 std::wcout, 119 #'0
#EQU1 std::wcout, 120 #'d
#EQU1 std::wcout, 101 #'1
#EQU1 std::wcout, 33 #'#
#EQU1 std::wcout, 34 #'-
#EQU1 std::wcout, 0 #'(null)
#D 96s, 121(#expr)

# Prints 100, hex 20, & 4 is for print string
#EQU1 std::wcout, 100, 4 # prints to the log
#EQU1 std::wendl

```

## (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

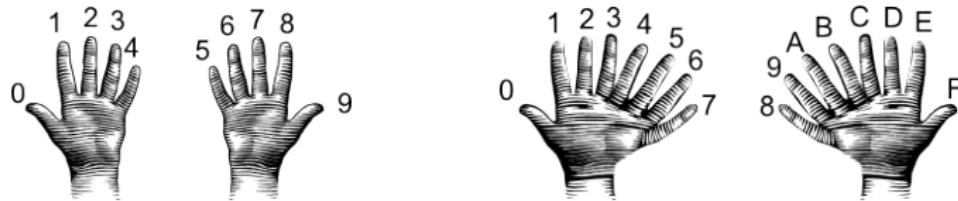


# Today's Topics



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

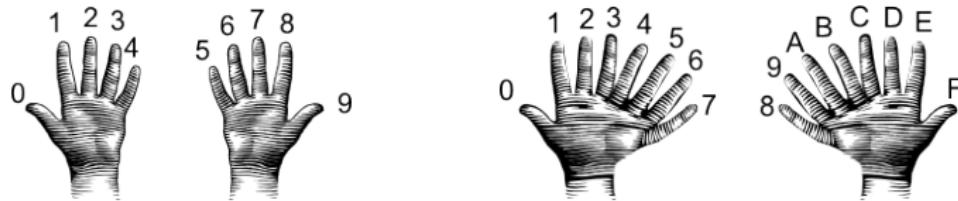
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

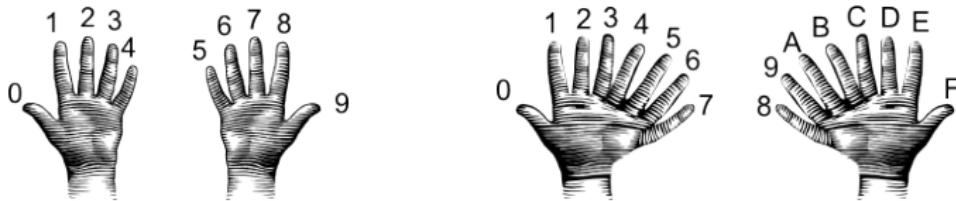
# Hexadecimal to Decimal: Converting Between Bases



(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.

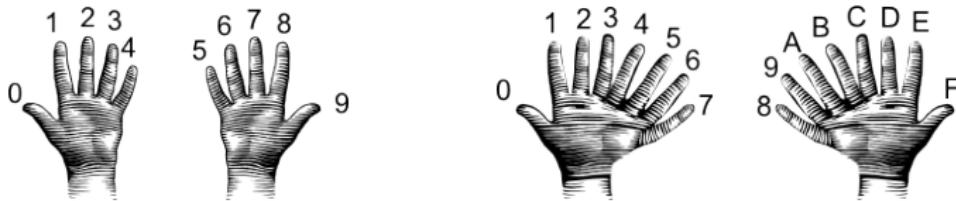
# Hexadecimal to Decimal: Converting Between Bases



(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?

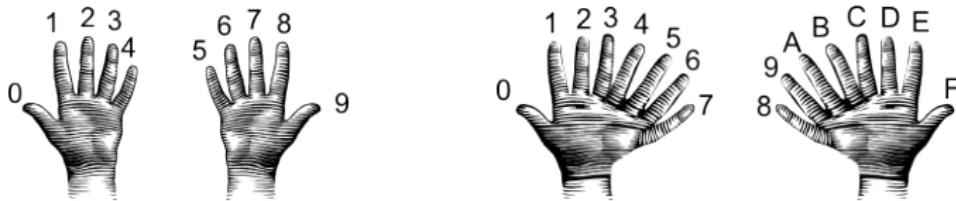
# Hexadecimal to Decimal: Converting Between Bases



(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.

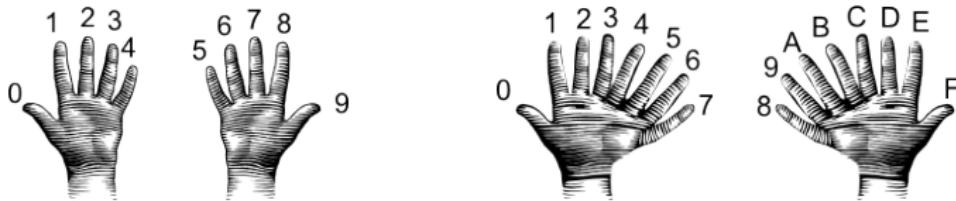
# Hexadecimal to Decimal: Converting Between Bases



(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.  $2 \times 16$  is 32.

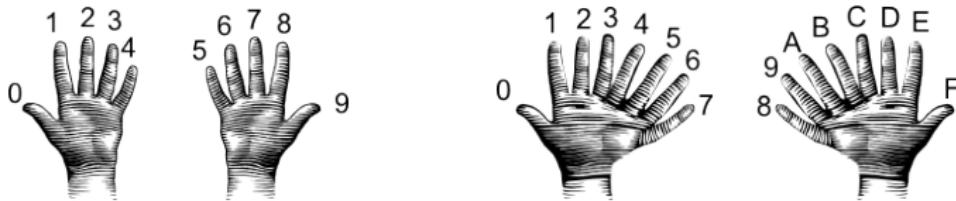
# Hexadecimal to Decimal: Converting Between Bases



(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.  $2 \times 16$  is 32.  
A in decimal digits is 10.

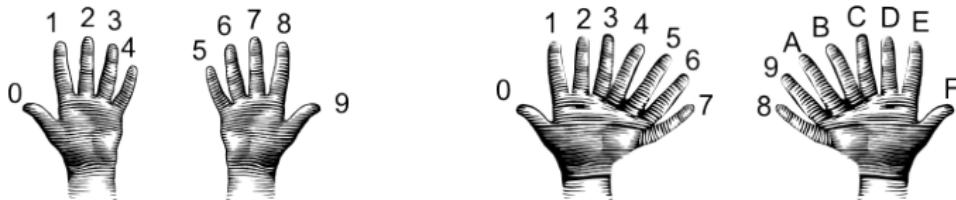
# Hexadecimal to Decimal: Converting Between Bases



(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.  $2 \times 16$  is 32.  
A in decimal digits is 10.  
 $32 + 10$  is 42.

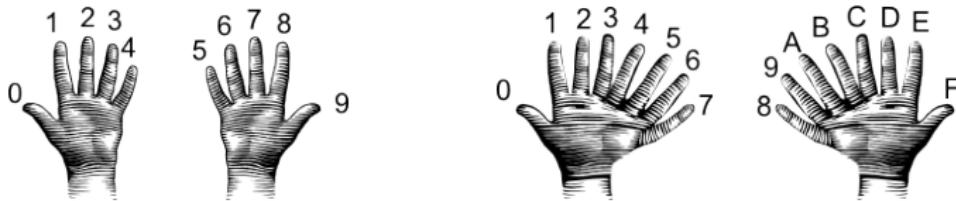
# Hexadecimal to Decimal: Converting Between Bases



(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.  $2 \times 16$  is 32.  
A in decimal digits is 10.  
 $32 + 10$  is 42.  
Answer is 42.
  - Example: what is 99 as a decimal number?

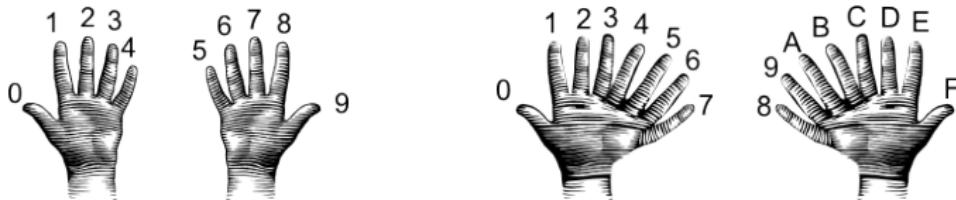
# Hexadecimal to Decimal: Converting Between Bases



(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.  $2 \times 16$  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.

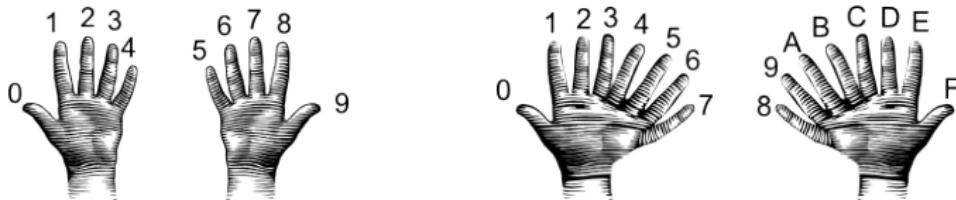
# Hexadecimal to Decimal: Converting Between Bases



(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.  $2 \times 16$  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.  $9 \times 16$  is 144.

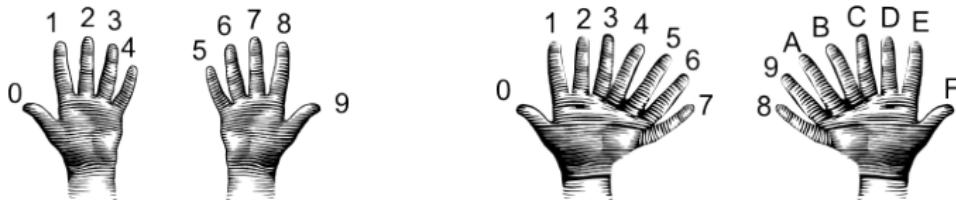
# Hexadecimal to Decimal: Converting Between Bases



(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.  $2 \times 16$  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.  $9 \times 16$  is 144.  
9 in decimal digits is 9

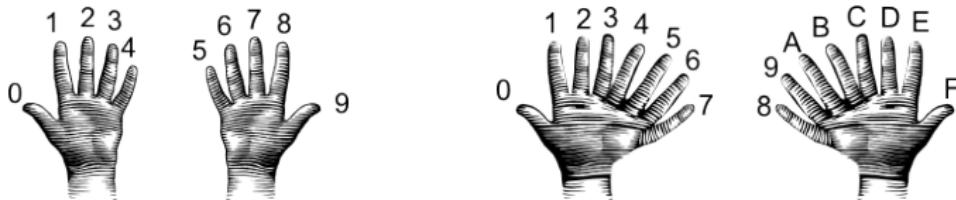
# Hexadecimal to Decimal: Converting Between Bases



(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.  $2 \times 16$  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.  $9 \times 16$  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.  $2 \times 16$  is 32.

A in decimal digits is 10.

$32 + 10$  is 42.

Answer is 42.

- Example: what is 99 as a decimal number?

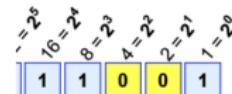
9 in decimal is 9.  $9 \times 16$  is 144.

9 in decimal digits is 9

$144 + 9$  is 153.

Answer is 153.

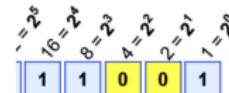
# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:
  - Divide by 128 ( $= 2^7$ ). Quotient is the first digit.

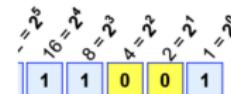
# Decimal to Binary: Converting Between Bases



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

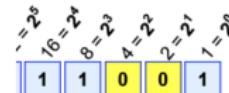
- From decimal to binary:
  - Divide by  $128 (= 2^7)$ . Quotient is the first digit.
  - Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.

# Decimal to Binary: Converting Between Bases



- 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.

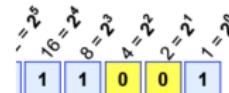
# Decimal to Binary: Converting Between Bases



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

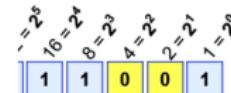


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

# Decimal to Binary: Converting Between Bases

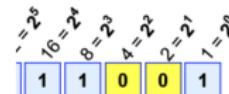


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

# Decimal to Binary: Converting Between Bases

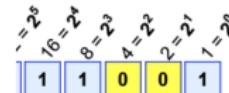


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

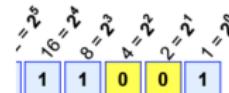
# Decimal to Binary: Converting Between Bases



- 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

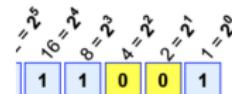


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



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

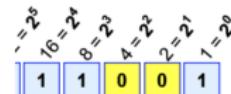
- From decimal to binary:

- 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.

# Decimal to Binary: Converting Between Bases



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

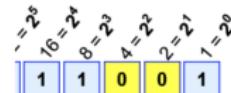
- From decimal to binary:

- 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:

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

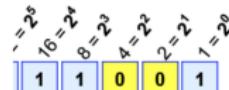
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

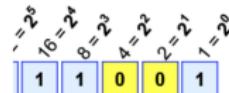
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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



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

- From decimal to binary:

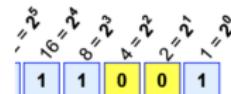
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

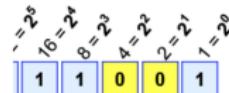
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

# 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.

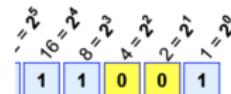
- Example: what is 130 in binary notation?

130/128 is 1 rem 2. 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:

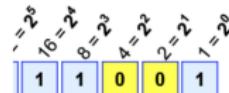
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

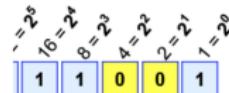
130/128 is 1 rem 2. First digit is 1: 1...

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?

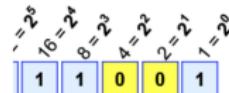
130/128 is 1 rem 2. First digit is 1: 1...

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



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?

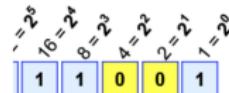
130/128 is 1 rem 2. First digit is 1: 1...

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...

# Decimal to Binary: Converting Between Bases



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

- 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...

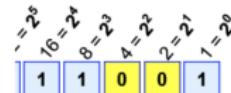
2/64 is 0 rem 2. Next digit is 0: 10...

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.

# 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...

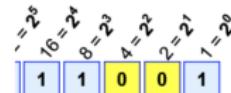
2/64 is 0 rem 2. Next digit is 0: 10...

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:

# 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...

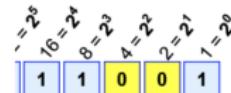
2/64 is 0 rem 2. Next digit is 0: 10...

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

- 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...

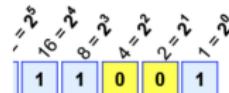
2/32 is 0 rem 2. Next digit is 0: 100...

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.

# Decimal to Binary: Converting Between Bases



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

- 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...

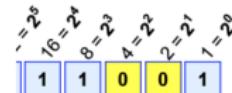
2/32 is 0 rem 2. Next digit is 0: 100...

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:

# 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...

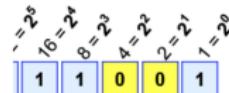
2/32 is 0 rem 2. Next digit is 0: 100...

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

- 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...

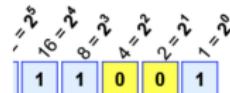
2/16 is 0 rem 2. Next digit is 0: 1000...

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.

# 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...

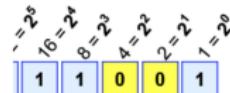
2/16 is 0 rem 2. Next digit is 0: 1000...

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:

# 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...

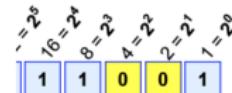
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

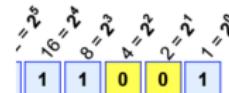
2/8 is 0 rem 2. Next digit is 0: 10000...

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...

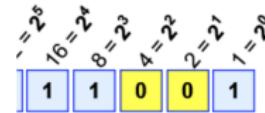
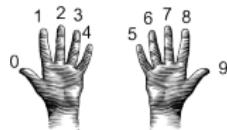
2/4 is 0 remainder 2. Next digit is 0: 100000...

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

Adding the last remainder: 10000010



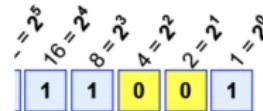
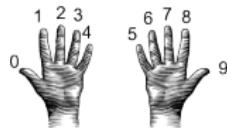
# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

# Decimal to Binary: Converting Between Bases

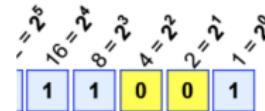
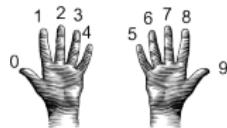


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99/128$  is 0 rem 99.

# Decimal to Binary: Converting Between Bases

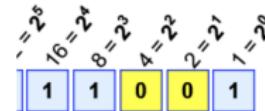
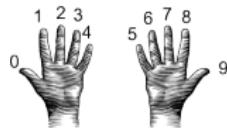


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

# Decimal to Binary: Converting Between Bases



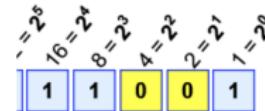
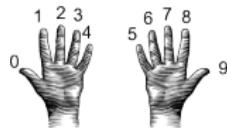
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



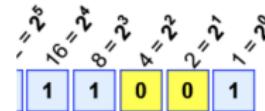
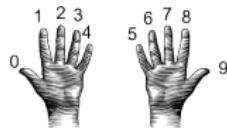
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



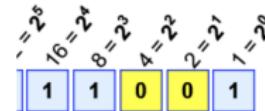
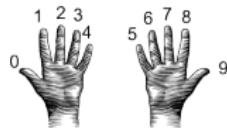
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

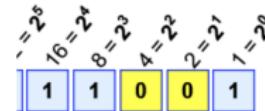
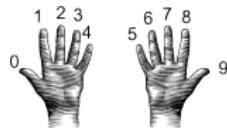
- Example: what is 99 in binary notation?

$99/128$  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.

# Decimal to Binary: Converting Between Bases



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

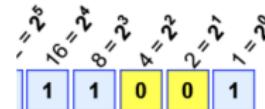
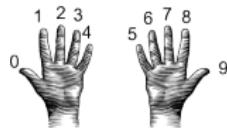
- Example: what is 99 in binary notation?

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

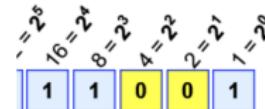
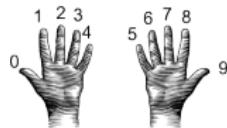
- Example: what is 99 in binary notation?

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

- Example: what is 99 in binary notation?

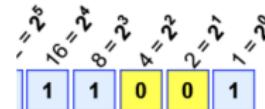
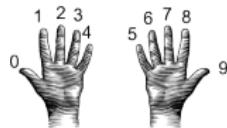
99/128 is 0 rem 99. First digit is 0: 0...

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.

# 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?

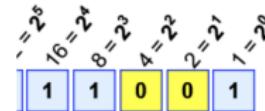
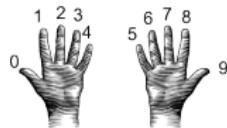
99/128 is 0 rem 99. First digit is 0: 0...

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?

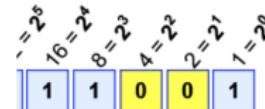
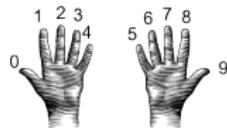
99/128 is 0 rem 99. First digit is 0: 0...

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...

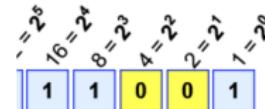
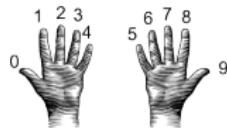
99/64 is 1 rem 35. Next digit is 1: 01...

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...

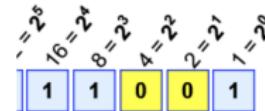
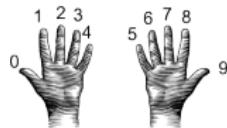
99/64 is 1 rem 35. Next digit is 1: 01...

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...

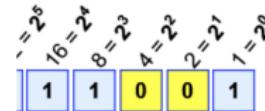
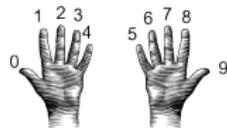
99/64 is 1 rem 35. Next digit is 1: 01...

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...

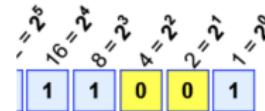
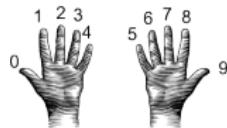
35/32 is 1 rem 3. Next digit is 1: 011...

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...

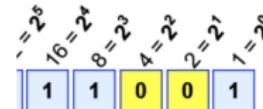
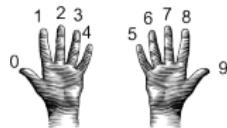
35/32 is 1 rem 3. Next digit is 1: 011...

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...

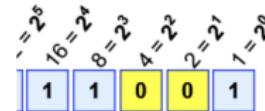
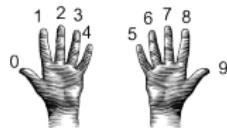
35/32 is 1 rem 3. Next digit is 1: 011...

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...

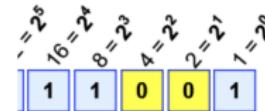
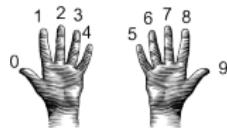
3/16 is 0 rem 3. Next digit is 0: 0110...

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:  $1 \times 16 + 1 \times 8 + 1 \times 0 + 1 \times 1 = 16 + 8 + 0 + 1 = 25$

- 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...

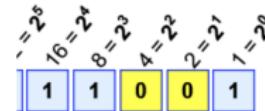
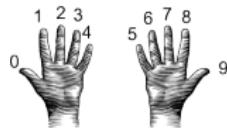
3/16 is 0 rem 3. Next digit is 0: 0110...

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...

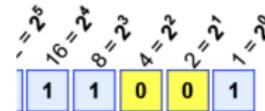
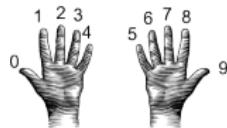
3/16 is 0 rem 3. Next digit is 0: 0110...

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

- 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...

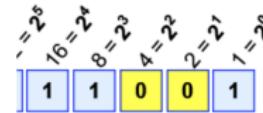
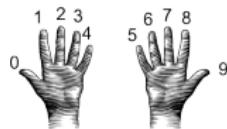
3/8 is 0 rem 3. Next digit is 0: 01100...

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



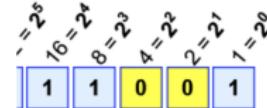
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

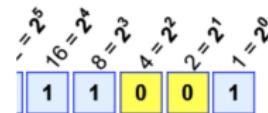
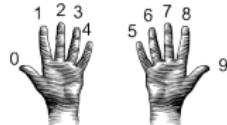
# Binary to Decimal: Converting Between Bases



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

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

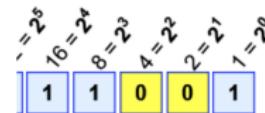
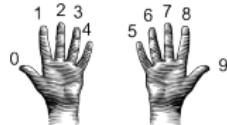
# Binary to Decimal: Converting Between Bases



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.

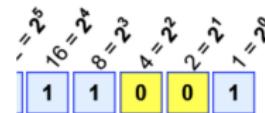
# Binary to Decimal: Converting Between Bases



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

- 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.

# Binary to Decimal: Converting Between Bases

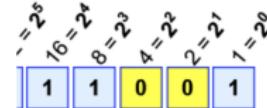
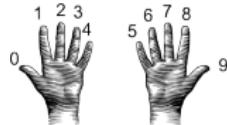


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

# Binary to Decimal: Converting Between Bases

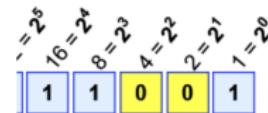
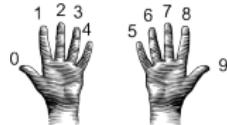


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

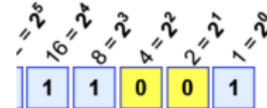
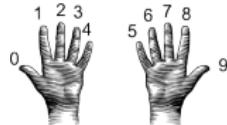


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

# Binary to Decimal: Converting Between Bases

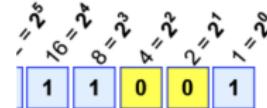


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

# Binary to Decimal: Converting Between Bases

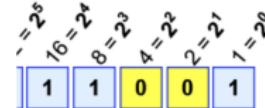


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

# Binary to Decimal: Converting Between Bases

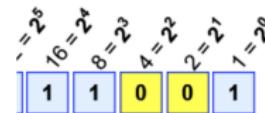


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

# Binary to Decimal: Converting Between Bases



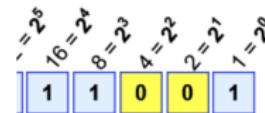
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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:

# Binary to Decimal: Converting Between Bases



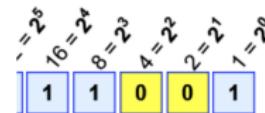
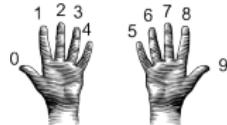
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

# Binary to Decimal: Converting Between Bases



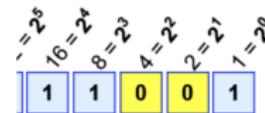
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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

# 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:

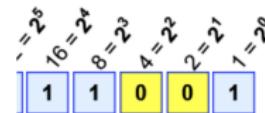
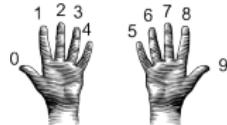
- Set sum = last digit.
- Multiply next digit by  $2^1$ . Add to sum.
- Multiply next digit by  $4 = 2^2$ . Add to sum.
- Multiply next digit by  $8 = 2^3$ . Add to sum.
- Multiply next digit by  $16 = 2^4$ . Add to sum.
- Multiply next digit by  $32 = 2^5$ . Add to sum.
- Multiply next digit by  $64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

# Binary to Decimal: Converting Between Bases



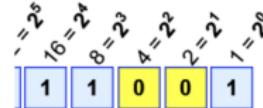
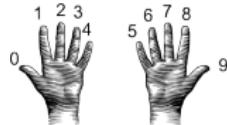
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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
$0 \times 2 = 0$ . Add 0 to sum:	1
$1 \times 4 = 4$ . Add 4 to sum:	5

# 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?

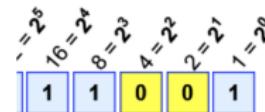
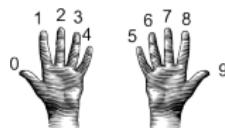
Sum starts with: 1

$0 \times 2 = 0$ . 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



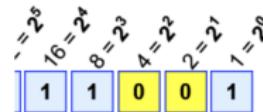
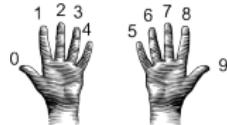
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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
$1 \times 8 = 8$ . Add 8 to sum:	13

# Binary to Decimal: Converting Between Bases



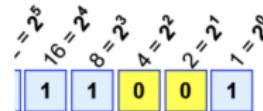
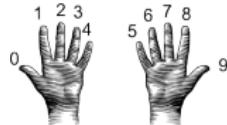
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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



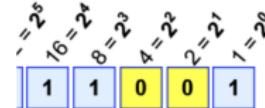
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

# Binary to Decimal: Converting Between Bases



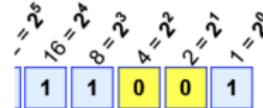
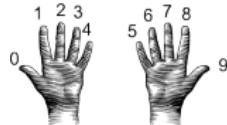
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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:

# Binary to Decimal: Converting Between Bases



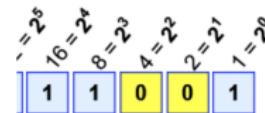
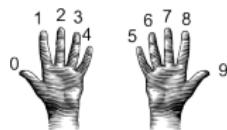
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

# Binary to Decimal: Converting Between Bases



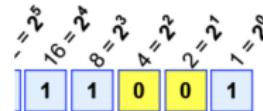
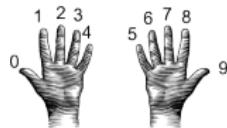
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

# Binary to Decimal: Converting Between Bases

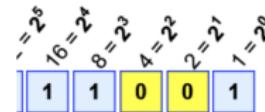
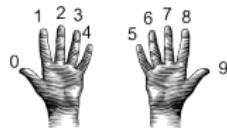


Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

# Binary to Decimal: Converting Between Bases



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

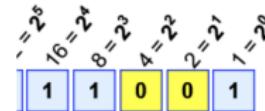
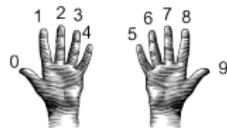
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

# Binary to Decimal: Converting Between Bases



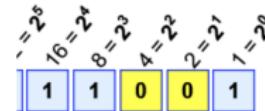
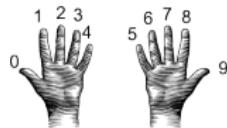
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

# Binary to Decimal: Converting Between Bases



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

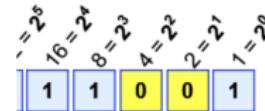
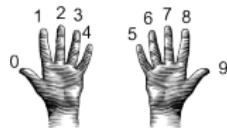
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

# Binary to Decimal: Converting Between Bases



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

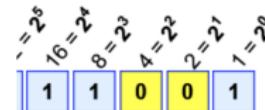
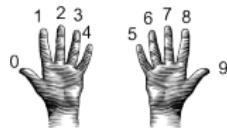
- Example: What is 10100100 in decimal?

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

- Example: What is 10100100 in decimal?

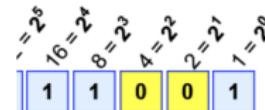
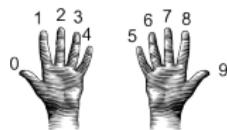
Sum starts with: 0

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

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

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

# Binary to Decimal: Converting Between Bases

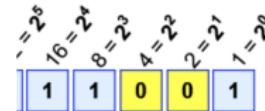
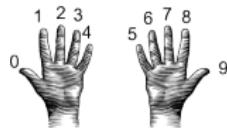


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

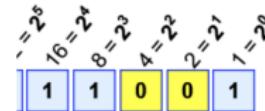
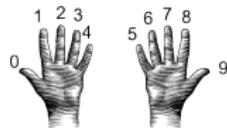
$0 \times 2 = 0$ . Add 0 to sum: 0

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

- Example: What is 10100100 in decimal?

Sum starts with: 0

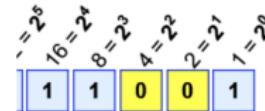
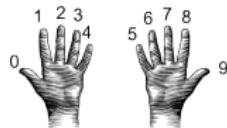
$0 \times 2 = 0$ . Add 0 to sum: 0

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

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

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

# 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

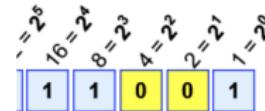
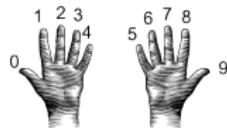
$1 \times 4 = 4$ . Add 4 to sum: 4

$0 \times 8 = 0$ . 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

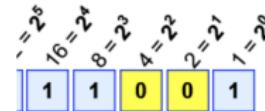
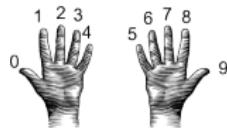


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

# 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

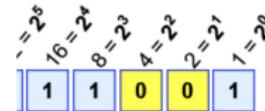
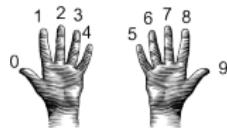
$0 \times 8 = 0$ . Add 0 to sum: 4

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

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

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

# 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

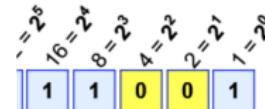
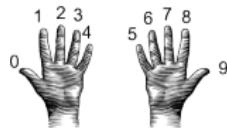
$0 \times 8 = 0$ . Add 0 to sum: 4

$0 \times 16 = 0$ . 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:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

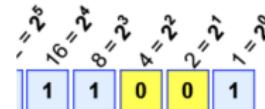
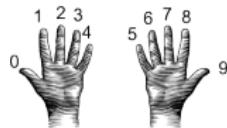
$0 \times 16 = 0$ . Add 0 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases

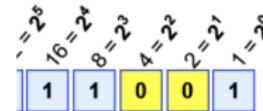
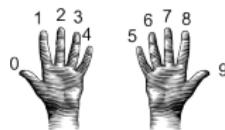


Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- 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

# Binary to Decimal: Converting Between Bases



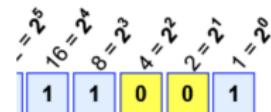
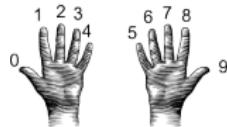
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

The answer is 164.

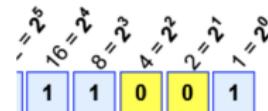
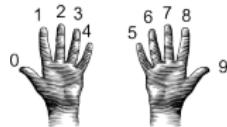
# Design Challenge: Incrementers



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

- Simplest arithmetic: add one ("increment") a variable.

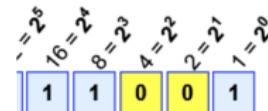
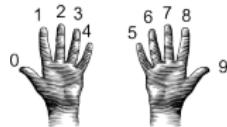
# Design Challenge: Incrementers



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

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

# Design Challenge: Incrementers

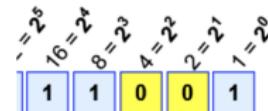
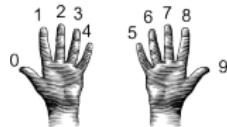


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Simplest arithmetic: add one (“increment”) a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

# Design Challenge: Incrementers



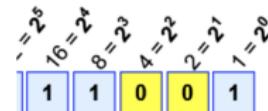
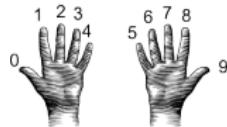
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.

# Design Challenge: Incrementers



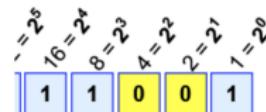
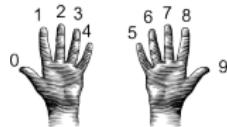
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.  
Example: "forty one" → "forty two"

# Design Challenge: Incrementers



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

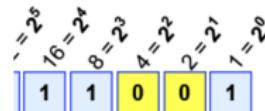
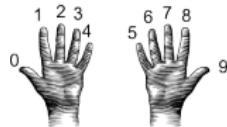
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.  
Example: "forty one" → "forty two"

*Hint: Convert to numbers, increment, and convert back to strings.*

# Design Challenge: Incrementers



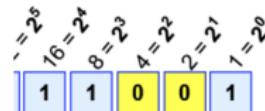
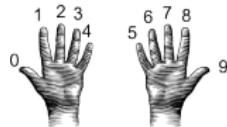
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.  
Example: "forty one" → "forty two"  
*Hint: Convert to numbers, increment, and convert back to strings.*
- Challenge: Write an algorithm for incrementing binary numbers.

# Design Challenge: Incrementers



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

- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):  
    m = n+1  
    return(m)
```

- Challenge: Write an algorithm for incrementing numbers expressed as words.  
Example: "forty one" → "forty two"

*Hint: Convert to numbers, increment, and convert back to strings.*

- Challenge: Write an algorithm for incrementing binary numbers.

Example: "1001" → "1010"

# 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.
- It is on paper. No use of computers, phones, etc. allowed.

# Final Overview: Format

- The exam is 2 hours long.
- 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.
- 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.
- 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.
  - ▶ Do not fold the paper; it's distracting to others taking the exam.

# Final Overview: Format

- The exam is 2 hours long.
- 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.
  - ▶ Do not fold the paper; it's distracting to others taking the exam.
  - ▶ Best if you design/write your own as it's an excellent way to study.

# Final Overview: Format

- The exam is 2 hours long.
- 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.
  - ▶ Do not fold the paper; it's distracting to others taking the exam.
  - ▶ Best if you design/write your own as it's an excellent way to study.
- The exam format:

# Final Overview: Format

- The exam is 2 hours long.
- 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.
  - ▶ Do not fold the paper; it's distracting to others taking the exam.
  - ▶ Best if you design/write your own as it's an excellent way to study.
- The exam format:
  - ▶ 10 questions, each worth 10 points.

# Final Overview: Format

- The exam is 2 hours long.
- 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.
  - ▶ Do not fold the paper; it's distracting to others taking the exam.
  - ▶ Best if you design/write your own as it's an 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.
- 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.
  - ▶ Do not fold the paper; it's distracting to others taking the exam.
  - ▶ Best if you design/write your own as it's an 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.
- 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.
  - ▶ Do not fold the paper; it's distracting to others taking the exam.
  - ▶ Best if you design/write your own as it's an 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.
- 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.
  - ▶ Do not fold the paper; it's distracting to others taking the exam.
  - ▶ Best if you design/write your own as it's an 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 the 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 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 respects all of academic dishonesty (e.g., plagiarism, cheating or commissing other学术不端, and fabrication of records and affidavits/documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and the Hunter College Academic Integrity Procedures.

I declare that all work is submitted honestly and is original to the best of my knowledge. All credit is given due.
Name:
Signature:
Date:
Signature:

# Exam Options

## Exam Times:

- Regular Time: Monday, May 22 in Assembly Hall, 9-11 am.

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 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 on examinations, submitting unfair advantages, and fabrication of records and affidavits/documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and the Hunter College Academic Integrity Procedures.

I declare that all work is submitted honestly and is original to the best of my knowledge. All credit is given due.
Name: _____
Signature: _____
Date: _____
Handwritten Signature: _____

# Exam Options

## Exam Times:

- **Regular Time: Monday, May 22 in Assembly Hall, 9-11 am.**
- **Alternate Time: Wednesday, May 17 in 1001G Hunter North, (time TBD).**

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 that you may 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 respects all of academic integrity (e.g., plagiarism, cheating or communication, sharing writer assignments, and fabrication of records and affidavits/documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and the Hunter College Academic Integrity Procedures.

I declare that all work is my own and I understand it will be reported to the Office of Student Life if all work is not mine.
Name: _____
Signature: _____
Date: _____
Witness: _____

# Exam Options

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 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 respects all acts of academic dishonesty (e.g., plagiarism, cheating or communication, sharing writer submissions, and fabrication of records and forged documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and the Hunter College Academic Integrity Policy and Procedures.

I declare that all work is submitted honestly and is original to the best of my knowledge. All credit is given where due.	
Name:	
Signature:	
Email:	
Phone:	

## Exam Times:

- **Regular Time: Monday, May 22 in Assembly Hall, 9-11 am.**
- **Alternate Time: Wednesday, May 17 in 1001G Hunter North, (time TBD).**
- **Survey for your exam date choice will be available next lecture.**

# Exam Options

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 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 respects all forms of academic dishonesty (e.g., plagiarism, cheating on examinations, submitting unfair advantages, and fabrication of records and affidavits documenting an excuse off from the subject of intellectual honesty). The College is committed to enforcing the CUNY Policy on Academic Dishonesty and the Hunter College Academic Integrity Policy. See the Hunter College Academic Integrity Procedures.

I declare that all work is submitted honestly and is original to the best of my knowledge. All credit is given where due.	
Name:	
Sugiharto	
Final:	
Homework:	

## Exam Times:

- **Regular Time: Monday, May 22 in Assembly Hall, 9-11 am.**
- **Alternate Time: Wednesday, May 17 in 1001G Hunter North, (time TBD).**
- Survey for your exam date choice will be available next lecture.
- If you choose to take the early date, **you will not be given access to the exam on May 22, even if you miss the early exam.**

# Weekly Reminders!



Before the next lecture, don't forget to:

- Work on this week's Online Lab

# Weekly Reminders!



Before the next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz

# Weekly Reminders!



Before the next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz
- Schedule an appointment to take the Code Review

# Weekly Reminders!



Before the next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz
- Schedule an appointment to take the Code Review
- Submit this week's programming assignments

# Weekly Reminders!



Before the next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz
- Schedule an appointment to take the Code Review
- Submit this week's programming assignments
- If you need help, schedule an appointment for Tutoring

# Weekly Reminders!



Before the next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz
- Schedule an appointment to take the Code Review
- Submit this week's programming assignments
- If you need help, schedule an appointment for Tutoring
- Take the Lecture Preview on Blackboard

# Lecture Slips & Writing Boards



- Hand your lecture slip to a UTA.
- Return writing boards as you leave.