

# CSci 127: Introduction to Computer Science



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

- This lecture will be recorded

# Announcements

- Thanksgiving Break starts in 9 days.



# Announcements

- Thanksgiving Break starts in 9 days.
- No CUNY classes:  
*Thursday-Saturday, 26-29 November.*



# Announcements



- Thanksgiving Break starts in 9 days.
- No CUNY classes:  
*Thursday-Saturday, 26-29 November.*
- Add my email to your contacts and check your spam folders. I reply within 24/48 hours at most (not on weekends).

# Announcements



- Thanksgiving Break starts in 9 days.
- No CUNY classes:  
*Thursday-Saturday, 26-29 November.*
- Add my email to your contacts and check your spam folders. I reply within 24/48 hours at most (not on weekends).
- In response to wrap-up requests, additional challenges today with while loops and binary & hexadecimal numbers.

# Frequently Asked Questions

From email and tutoring.

- **Help! Binary & hexadecimal numbers make no sense!**

# Frequently Asked Questions

From email and tutoring.

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

# Frequently Asked Questions

From email and tutoring.

- **Help! Binary & hexadecimal numbers make no sense!**  
*No worries— we'll start off with those in today's lecture.*
- **When is the final? Is there a review sheet?**

# Frequently Asked Questions

From email and tutoring.

- **Help! Binary & hexadecimal numbers make no sense!**  
*No worries— we'll start off with those in today's lecture.*
- **When is the final? Is there a review sheet?**  
*The official final is Monday, 14 December, 9-11am.*

# Frequently Asked Questions

From email and tutoring.

- **Help! Binary & hexadecimal numbers make no sense!**

*No worries— we'll start off with those in today's lecture.*

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

*The official final is Monday, 14 December, 9-11am.*

*The early final exam (alternative date) is on Friday, 11 December (Reading Day), 8-10am.*

# Frequently Asked Questions

From email and tutoring.

- **Help! Binary & hexadecimal numbers make no sense!**

*No worries— we'll start off with those in today's lecture.*

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

*The official final is Monday, 14 December, 9-11am.*

*The early final exam (alternative date) is on Friday, 11 December (Reading Day), 8-10am.*

*Instead of a review sheet, we have:*

# Frequently Asked Questions

From email and tutoring.

- **Help! Binary & hexadecimal numbers make no sense!**

*No worries— we'll start off with those in today's lecture.*

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

*The official final is Monday, 14 December, 9-11am.*

*The early final exam (alternative date) is on Friday, 11 December (Reading Day), 8-10am.*

*Instead of a review sheet, we have:*

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

# Frequently Asked Questions

From email and tutoring.

- **Help! Binary & hexadecimal numbers make no sense!**

*No worries— we'll start off with those in today's lecture.*

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

*The official final is Monday, 14 December, 9-11am.*

*The early final exam (alternative date) is on Friday, 11 December (Reading Day), 8-10am.*

*Instead of a review sheet, we have:*

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

# Frequently Asked Questions

From email and tutoring.

- **Help! Binary & hexadecimal numbers make no sense!**

*No worries— we'll start off with those in today's lecture.*

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

*The official final is Monday, 14 December, 9-11am.*

*The early final exam (alternative date) is on Friday, 11 December (Reading Day), 8-10am.*

*Instead of a review sheet, we have:*

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *UTAs in drop-in tutoring happy to review concepts and old exam questions.*
- ▶ *There will be opportunity for some practice and to ask review questions during our last meeting on 8 December.*

# Today's Topics



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

# Today's Topics



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

# Predict what the code will do:

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

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

# Python Tutor

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

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

(Demo with pythonTutor)

# Design Pattern: Linear Search

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

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

- Example of **linear search**.

# Design Pattern: Linear Search

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

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

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

# Design Pattern: Linear Search

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

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

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

# Design Pattern: Linear Search

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

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

- Example of **linear search**.
- Start at the beginning of the list.
- Look at each item, one-by-one.
- Stopping, 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
- Final Exam: Format

# Python & Circuits Review: 10 Weeks in 10 Minutes



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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

- As well as definite loops & the turtle package:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

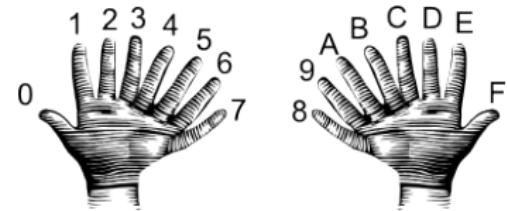
## Week 2: variables, data types, more on loops & range()

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

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

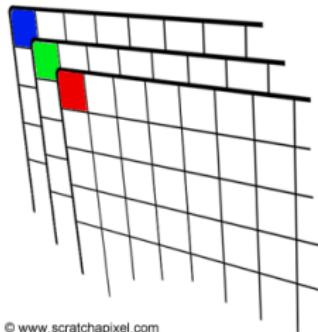
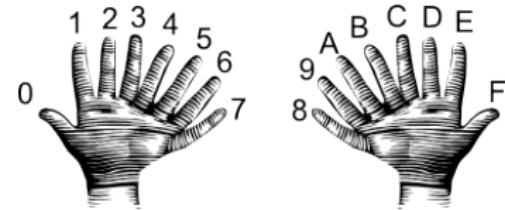
# Week 3: colors, hex, slices, numpy & images

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



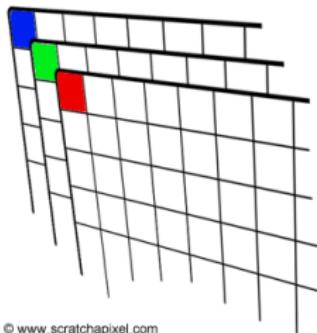
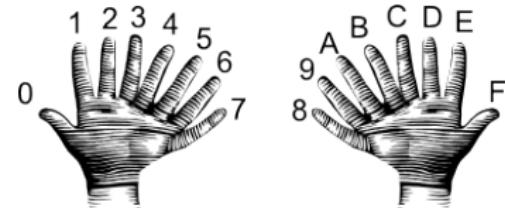
# Week 3: colors, hex, slices, numpy & images

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



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

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



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

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

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

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

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

# Week 4: design problem (cropping images) & decisions



# Week 4: design problem (cropping images) & decisions



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

# Week 4: design problem (cropping images) & decisions



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

# Week 4: design problem (cropping images) & decisions



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

## Week 4: design problem (cropping images) & decisions

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

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

# Week 5: logical operators, truth tables & logical circuits

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

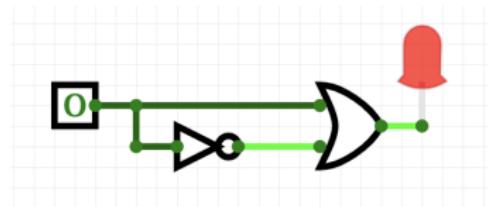
visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

# Week 5: logical operators, truth tables & logical circuits

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

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

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



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

Source: [https://en.wikipedia.org/wiki/Demographics\\_of\\_New\\_York\\_City](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,Brooklyn,Queens,Bronx,Staten Island,Total  
1890,1,203,727,718,1,203  
1891,2,184,3,462,,2,847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,63545,5830,6442,1755,4543,75934  
1820,123704,11487,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,313110,12013,14043,5348,10965,391114  
1850,35344,12800,18951,5348,10965,391115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911801  
1890,1367711,707128,67243,51980,33029,2141314  
1900,185093,116582,152999,200567,67921,2437202  
1910,2233142,1634351,284041,430980,8569,4766803  
1920,2211103,2018354,446031,722013,116582,591048  
1930,1867128,2203936,1191325,1158243,7071645,4930446  
1940,1889924,2499285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7891957  
1960,1696010,2738175,1809949,1451277,191555,7891984  
1970,1539231,2460705,1471705,1471705,135443,7891984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,447512,8175133  
2015,1444518,2436733,2339150,1459444,474558,8059405

nycHistPop.csv

In Lab 6

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

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

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,,727,7881,28423  
1870,33131,4549,6159,1781,3827,49447  
1860,60515,5740,6442,1755,4543,75955  
1850,55545,5254,5851,1541,3973,74934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14031,5348,10965,391114  
1850,35545,21891,18591,5348,10965,391114  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59945,5653,51980,33029,1911803  
1890,1367711,707128,65441,51861,31861,2151134  
1900,185093,116582,152999,200567,67621,2437202  
1910,233142,1634351,284041,430980,8569,4766803  
1920,2210103,2018354,446071,446071,73201,11651,50048  
1930,1867137,1796128,205254,205254,5583,4930446  
1940,1889924,2469285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690131,205254,189091,1451277,191555,7981984  
1970,1539231,1865701,1471701,1471701,135443,7981984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,379977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2216722,1385108,4175133,8175133  
2015,1444018,2436733,2339150,1459444,474558,8059405

nycHistPop.csv

In Lab 6

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

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Borough,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,2017,...,727,7181  
1771,21843,36232,...,2847,28423  
1790,33131,4549,...,6159,1781,3827,49447  
1800,60515,5740,...,6442,1755,4543,75955  
1810,67500,6200,...,6442,1755,4543,75934  
1820,123704,11187,...,8246,2792,6135,152056  
1830,20589,20535,...,9049,3023,7082,242278  
1840,31210,21013,...,14093,5346,10965,391114  
1850,35549,21800,...,18591,5346,10965,391115  
1860,51349,...,279122,...,23993,...,25492,174777  
1870,942292,...,419921,...,45468,...,37393,...,33029,...,1479103  
1880,1164473,...,59943,...,5653,...,51980,...,39301,...,1911801  
1890,1364473,...,65943,...,5653,...,51980,...,39301,...,1911804  
1900,185093,...,116582,...,152999,...,200567,...,67921,...,2437202  
1910,2233142,...,1634351,...,2841,...,430980,...,8569,...,476683  
1920,22161103,...,2018354,...,44601,...,72021,...,11651,...,593083  
1930,26671103,...,2018354,...,44601,...,72021,...,11651,...,5930446  
1940,...,1889924,...,2690285,...,1297634,...,1394711,...,174441,...,7454995  
1950,...,1960101,...,2738175,...,1550949,...,1451277,...,191555,...,7991957  
1960,...,1660101,...,2738175,...,1550949,...,1451277,...,191555,...,7991984  
1970,...,1660101,...,2738175,...,1550949,...,1451277,...,191555,...,7991984  
1980,...,1426285,...,2230936,...,1891325,...,1168972,...,352121,...,7071639  
1990,...,1487536,...,2300664,...,1951598,...,1302789,...,379977,...,7322564  
2000,...,1537195,...,2485326,...,2229379,...,1332650,...,419728,...,8080879  
2010,...,1583873,...,2504705,...,2272722,...,1385108,...,474558,...,8175133  
2015,...,1444918,...,2546733,...,2339150,...,1459446,...,474558,...,8159405
```

nycHistPop.csv

In Lab 6

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

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Population  
1690,203,2037,...,727,7181  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,6350,7000,1800,5300,93734  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31510,19113,14000,5348,10965,391114  
1850,35549,21890,18850,5800,12000,501115  
1860,813469,279122,23903,23933,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1385000,720000,68000,51800,35000,210000  
1900,1850093,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44600,720201,11650,50000  
1930,26671128,2203936,1796128,235428,5820,4930446  
1940,1889924,2690285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690000,2319319,1890000,1460000,191555,7981984  
1970,1539231,2460701,1871473,1472701,193443,7984640  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,379877,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,474558,8175133  
2015,1444018,2640733,2339150,1459446,474558,8056405
```

nycHistPop.csv

In Lab 6

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

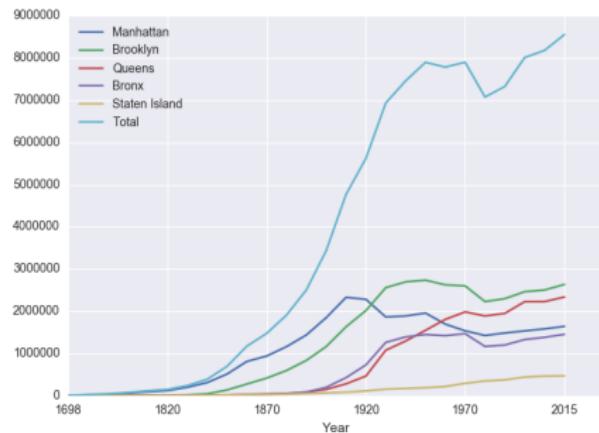
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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  
1771,21843,36231,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,71531,6349,7041,1813,5197,93734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,19113,14071,5348,10965,391114  
1850,355441,21861,18951,6851,12501,5115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33051,1911801  
1890,1380000,718000,68000,55000,38000,210000  
1900,1850093,116582,152999,200567,67921,2437202  
1910,233142,1634351,2841,430980,8569,476683  
1920,2210103,2018354,44607,73201,11651,50048  
1930,2667103,2486128,479128,52545,5830,4930446  
1940,1889924,2690285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7091957  
1960,1690000,2319319,1809000,1600000,1200000,781984  
1970,1539231,2465701,1472701,1235443,708462  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1497536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2216722,1385108,451721,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6



# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 8: function parameters, github

- Functions can have **input parameters**.

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

# Week 8: function parameters, github

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

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip: '))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip: '))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

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

# Week 8: function parameters, github

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

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip: '))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip: '))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

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

# Week 8: function parameters, github

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

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip: '))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip: '))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

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

# Week 8: function parameters, github

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

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip: '))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip: '))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

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

# Week 8: function parameters, github

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

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip: '))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)
                                Actual Parameters

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip: '))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

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

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



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

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

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

print('The distance entered is', dist)
```

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

```
import turtle
import random

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

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

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

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

print('The distance entered is', dist)
```

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

```
import turtle
import random

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

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

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

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

print('The distance entered is', dist)
```

```
import turtle
import random

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

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

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

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

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

print('The distance entered is', dist)
```

```
import turtle
import random

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

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

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

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

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

print('The distance entered is', dist)
```

```
import turtle
import random

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

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

- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.
- Very useful for checking user input for correctness.
- Python's built-in random package has useful methods for generating random whole numbers and real numbers.
- To use, must include:  
`import random`.
- The max design pattern provides a template for finding maximum value from a list.

# Python & Circuits Review: 10 Weeks in 10 Minutes



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

# Lecture Quiz

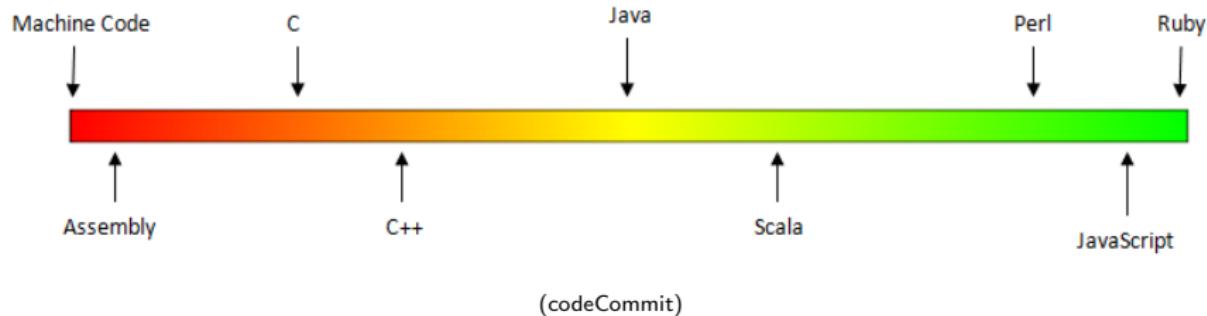
- Log-in to Gradescope
- Find LECTURE 11 Quiz
- Take the quiz
- **You have 3 minutes**

# Today's Topics



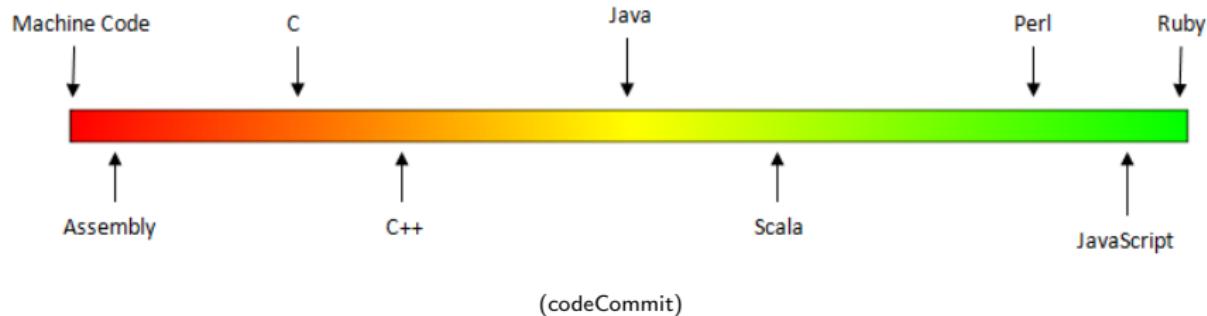
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

# Low-Level vs. High-Level Languages



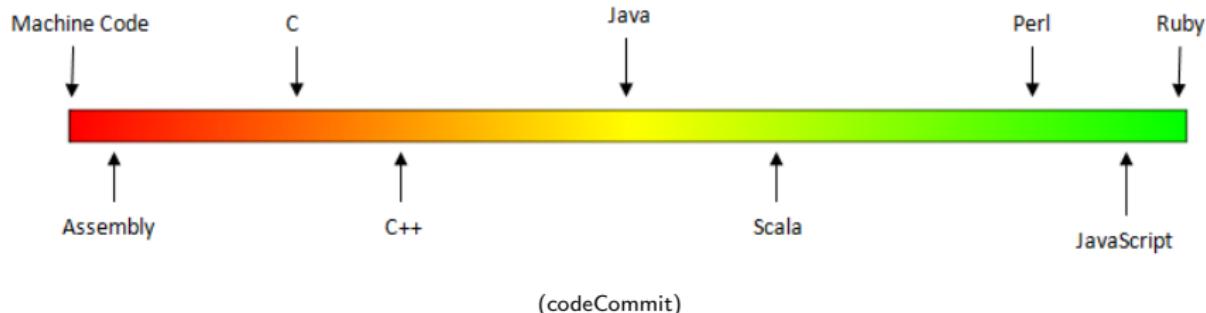
- Can view programming languages on a continuum.

# Low-Level vs. High-Level Languages



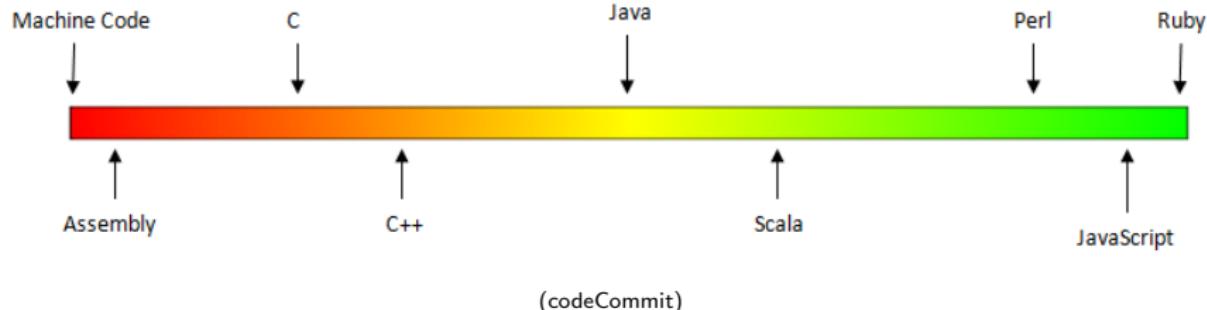
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

# Low-Level vs. High-Level Languages



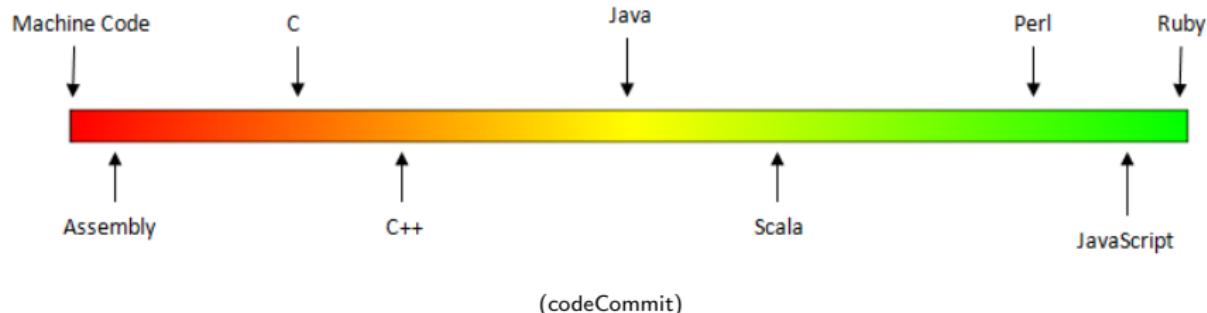
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

# 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



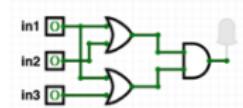
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine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

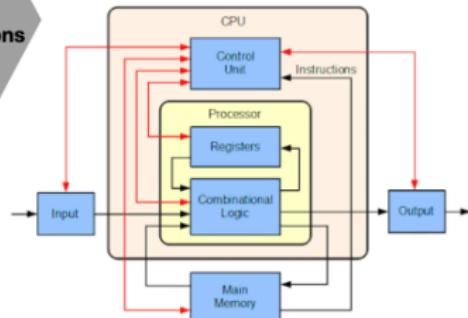
Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.  
Dies ist ein Blindtext. An ihm lässt sich



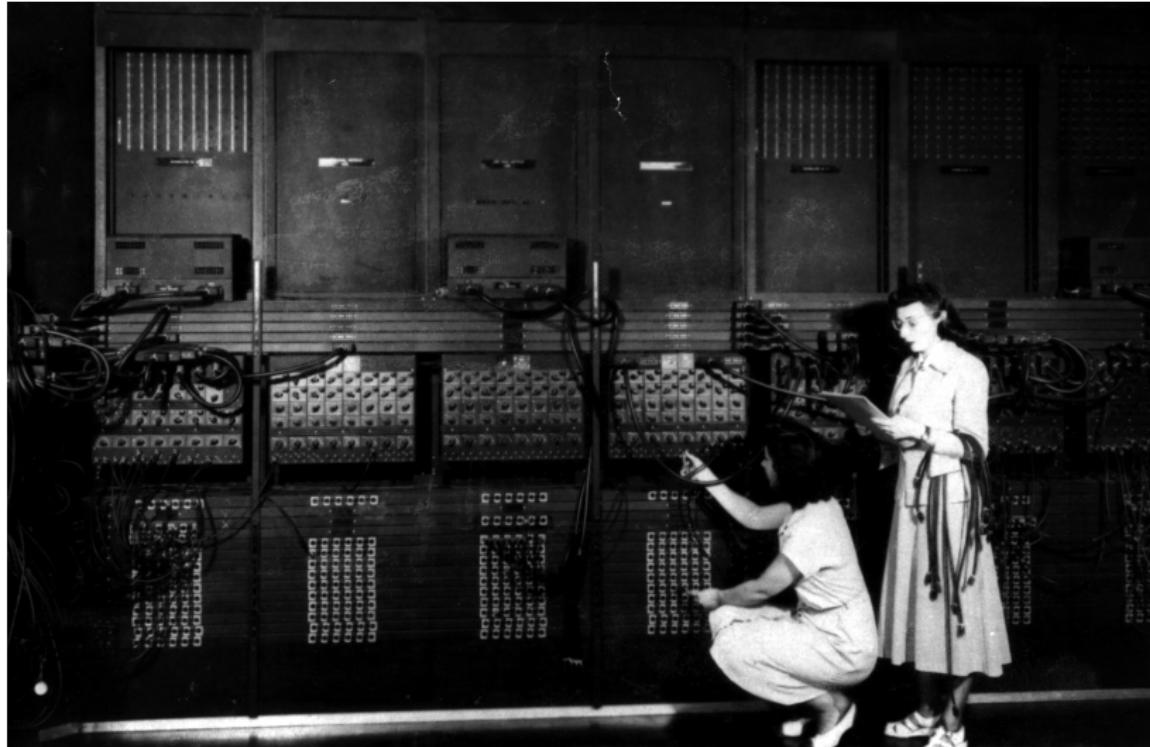
```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)
```



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



# 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:UU .....
```

(wiki)

# Machine Language

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

(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.

(wiki)

# Machine Language



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

Assembly Instruction	Binary Value
LDI R0, 1000	00 0000 C0 30
LDI R1, 1000	00 0000 C0 30
ADD R0, R1, R0	00 0002 1B
ADD R0, R1, R0	00 0002 1B
LDI R2, 0	00 0003 FB
LDI R2, 0	00 0003 FB
SEI	00 0004 0C
SEI	00 0004 0C
LDI R3, 1000	00 0005 34 12
LDI R3, 1000	00 0005 34 12
ADC R0, R3, R0	00 0006 69 21 43
ADC R0, R3, R0	00 0006 69 21 43
LDI R4, 1000	00 0007 8F 03
LDI R4, 1000	00 0007 8F 03
SUB R0, R4, R0	00 0008 8F 03
SUB R0, R4, R0	00 0008 8F 03
MUL R0, R0, R0	00 0009 E2 30
MUL R0, R0, R0	00 0009 E2 30
SEI	00 000A 98
SEI	00 000A 98
RET	00 000B 98
RET	00 000B 98
BREAK	00 000C 7F03
BREAK	00 000C 7F03
00 000D 0000 0000 0000 0000 0000 0000	00 000D 0000 0000 0000 0000 0000 0000
00 000E 0000 0000 0000 0000 0000 0000	00 000E 0000 0000 0000 0000 0000 0000
00 000F 0000 0000 0000 0000 0000 0000	00 000F 0000 0000 0000 0000 0000 0000
00 0010 0000 0000 0000 0000 0000 0000	00 0010 0000 0000 0000 0000 0000 0000
00 0011 0000 0000 0000 0000 0000 0000	00 0011 0000 0000 0000 0000 0000 0000
00 0012 0000 0000 0000 0000 0000 0000	00 0012 0000 0000 0000 0000 0000 0000

(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



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

Assembly Instruction	Binary Value
LDI R0, 1000	00000000 00000000
ADD R0, R1, R2	00000001 00000000
SUB R0, R1, R2	00000002 00000000
MUL R0, R1, R2	00000003 00000000

(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

## (Demo with WeMIPS)

## MIPS Commands

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

## 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

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

```
1 # Shows "Hello world" at the top of the stack
2
3 .text
4 .globl _start
5
6 .data
7 _str: .asciz "Hello world\n"
8
9 .text
10 _start:
11     li $t0, _str
12     li $t1, 11
13     add $t2, $t0, $t1
14     li $t3, 10
15     add $t4, $t2, $t3
16     li $t5, 10
17     add $t6, $t4, $t5
18     li $t7, 10
19     add $t8, $t6, $t7
20     li $t9, 10
21     add $t10, $t8, $t9
22     li $t11, 10
23     add $t12, $t10, $t11
24     li $t13, 10
25     add $t14, $t12, $t13
26     li $t15, 10
27     add $t16, $t14, $t15
28     li $t17, 10
29     add $t18, $t16, $t17
30     li $t19, 10
31     add $t20, $t18, $t19
32     li $t21, 10
33     add $t22, $t20, $t21
34     li $t23, 10
35     add $t24, $t22, $t23
36     li $t25, 10
37     add $t26, $t24, $t25
38     li $t27, 10
39     add $t28, $t26, $t27
40     li $t29, 10
41     add $t30, $t28, $t29
42     li $t31, 10
43     add $t32, $t30, $t31
44     li $t33, 10
45     add $t34, $t32, $t33
46     li $t35, 10
47     add $t36, $t34, $t35
48     li $t37, 10
49     add $t38, $t36, $t37
50     li $t39, 10
51     add $t40, $t38, $t39
52     li $t41, 10
53     add $t42, $t40, $t41
54     li $t43, 10
55     add $t44, $t42, $t43
56     li $t45, 10
57     add $t46, $t44, $t45
58     li $t47, 10
59     add $t48, $t46, $t47
60     li $t49, 10
61     add $t50, $t48, $t49
62     li $t51, 10
63     add $t52, $t50, $t51
64     li $t53, 10
65     add $t54, $t52, $t53
66     li $t55, 10
67     add $t56, $t54, $t55
68     li $t57, 10
69     add $t58, $t56, $t57
70     li $t59, 10
71     add $t60, $t58, $t59
72     li $t61, 10
73     add $t62, $t60, $t61
74     li $t63, 10
75     add $t64, $t62, $t63
76     li $t65, 10
77     add $t66, $t64, $t65
78     li $t67, 10
79     add $t68, $t66, $t67
80     li $t69, 10
81     add $t70, $t68, $t69
82     li $t71, 10
83     add $t72, $t70, $t71
84     li $t73, 10
85     add $t74, $t72, $t73
86     li $t75, 10
87     add $t76, $t74, $t75
88     li $t77, 10
89     add $t78, $t76, $t77
90     li $t79, 10
91     add $t80, $t78, $t79
92     li $t81, 10
93     add $t82, $t80, $t81
94     li $t83, 10
95     add $t84, $t82, $t83
96     li $t85, 10
97     add $t86, $t84, $t85
98     li $t87, 10
99     add $t88, $t86, $t87
100    li $t89, 10
101    add $t90, $t88, $t89
102    li $t91, 10
103    add $t92, $t90, $t91
104    li $t93, 10
105    add $t94, $t92, $t93
106    li $t95, 10
107    add $t96, $t94, $t95
108    li $t97, 10
109    add $t98, $t96, $t97
110    li $t99, 10
111    add $t100, $t98, $t99
112    li $t101, 10
113    add $t102, $t100, $t101
114    li $t103, 10
115    add $t104, $t102, $t103
116    li $t105, 10
117    add $t106, $t104, $t105
118    li $t107, 10
119    add $t108, $t106, $t107
120    li $t109, 10
121    add $t110, $t108, $t109
122    li $t111, 10
123    add $t112, $t110, $t111
124    li $t113, 10
125    add $t114, $t112, $t113
126    li $t115, 10
127    add $t116, $t114, $t115
128    li $t117, 10
129    add $t118, $t116, $t117
130    li $t119, 10
131    add $t120, $t118, $t119
132    li $t121, 10
133    add $t122, $t120, $t121
134    li $t123, 10
135    add $t124, $t122, $t123
136    li $t125, 10
137    add $t126, $t124, $t125
138    li $t127, 10
139    add $t128, $t126, $t127
140    li $t129, 10
141    add $t130, $t128, $t129
142    li $t131, 10
143    add $t132, $t130, $t131
144    li $t133, 10
145    add $t134, $t132, $t133
146    li $t135, 10
147    add $t136, $t134, $t135
148    li $t137, 10
149    add $t138, $t136, $t137
150    li $t139, 10
151    add $t140, $t138, $t139
152    li $t141, 10
153    add $t142, $t140, $t141
154    li $t143, 10
155    add $t144, $t142, $t143
156    li $t145, 10
157    add $t146, $t144, $t145
158    li $t147, 10
159    add $t148, $t146, $t147
160    li $t149, 10
161    add $t150, $t148, $t149
162    li $t151, 10
163    add $t152, $t150, $t151
164    li $t153, 10
165    add $t154, $t152, $t153
166    li $t155, 10
167    add $t156, $t154, $t155
168    li $t157, 10
169    add $t158, $t156, $t157
170    li $t159, 10
171    add $t160, $t158, $t159
172    li $t161, 10
173    add $t162, $t160, $t161
174    li $t163, 10
175    add $t164, $t162, $t163
176    li $t165, 10
177    add $t166, $t164, $t165
178    li $t167, 10
179    add $t168, $t166, $t167
180    li $t169, 10
181    add $t170, $t168, $t169
182    li $t171, 10
183    add $t172, $t170, $t171
184    li $t173, 10
185    add $t174, $t172, $t173
186    li $t175, 10
187    add $t176, $t174, $t175
188    li $t177, 10
189    add $t178, $t176, $t177
190    li $t179, 10
191    add $t180, $t178, $t179
192    li $t181, 10
193    add $t182, $t180, $t181
194    li $t183, 10
195    add $t184, $t182, $t183
196    li $t185, 10
197    add $t186, $t184, $t185
198    li $t187, 10
199    add $t188, $t186, $t187
200    li $t189, 10
201    add $t190, $t188, $t189
202    li $t191, 10
203    add $t192, $t190, $t191
204    li $t193, 10
205    add $t194, $t192, $t193
206    li $t195, 10
207    add $t196, $t194, $t195
208    li $t197, 10
209    add $t198, $t196, $t197
210    li $t199, 10
211    add $t200, $t198, $t199
212    li $t201, 10
213    add $t202, $t200, $t201
214    li $t203, 10
215    add $t204, $t202, $t203
216    li $t205, 10
217    add $t206, $t204, $t205
218    li $t207, 10
219    add $t208, $t206, $t207
220    li $t209, 10
221    add $t210, $t208, $t209
222    li $t211, 10
223    add $t212, $t210, $t211
224    li $t213, 10
225    add $t214, $t212, $t213
226    li $t215, 10
227    add $t216, $t214, $t215
228    li $t217, 10
229    add $t218, $t216, $t217
230    li $t219, 10
231    add $t220, $t218, $t219
232    li $t221, 10
233    add $t222, $t220, $t221
234    li $t223, 10
235    add $t224, $t222, $t223
236    li $t225, 10
237    add $t226, $t224, $t225
238    li $t227, 10
239    add $t228, $t226, $t227
240    li $t229, 10
241    add $t230, $t228, $t229
242    li $t231, 10
243    add $t232, $t230, $t231
244    li $t233, 10
245    add $t234, $t232, $t233
246    li $t235, 10
247    add $t236, $t234, $t235
248    li $t237, 10
249    add $t238, $t236, $t237
250    li $t239, 10
251    add $t240, $t238, $t239
252    li $t241, 10
253    add $t242, $t240, $t241
254    li $t243, 10
255    add $t244, $t242, $t243
256    li $t245, 10
257    add $t246, $t244, $t245
258    li $t247, 10
259    add $t248, $t246, $t247
260    li $t249, 10
261    add $t250, $t248, $t249
262    li $t251, 10
263    add $t252, $t250, $t251
264    li $t253, 10
265    add $t254, $t252, $t253
266    li $t255, 10
267    add $t256, $t254, $t255
268    li $t257, 10
269    add $t258, $t256, $t257
270    li $t259, 10
271    add $t260, $t258, $t259
272    li $t261, 10
273    add $t262, $t260, $t261
274    li $t263, 10
275    add $t264, $t262, $t263
276    li $t265, 10
277    add $t266, $t264, $t265
278    li $t267, 10
279    add $t268, $t266, $t267
280    li $t269, 10
281    add $t270, $t268, $t269
282    li $t271, 10
283    add $t272, $t270, $t271
284    li $t273, 10
285    add $t274, $t272, $t273
286    li $t275, 10
287    add $t276, $t274, $t275
288    li $t277, 10
289    add $t278, $t276, $t277
290    li $t279, 10
291    add $t280, $t278, $t279
292    li $t281, 10
293    add $t282, $t280, $t281
294    li $t283, 10
295    add $t284, $t282, $t283
296    li $t285, 10
297    add $t286, $t284, $t285
298    li $t287, 10
299    add $t288, $t286, $t287
300    li $t289, 10
301    add $t290, $t288, $t289
302    li $t291, 10
303    add $t292, $t290, $t291
304    li $t293, 10
305    add $t294, $t292, $t293
306    li $t295, 10
307    add $t296, $t294, $t295
308    li $t297, 10
309    add $t298, $t296, $t297
310    li $t299, 10
311    add $t300, $t298, $t299
312    li $t301, 10
313    add $t302, $t300, $t301
314    li $t303, 10
315    add $t304, $t302, $t303
316    li $t305, 10
317    add $t306, $t304, $t305
318    li $t307, 10
319    add $t308, $t306, $t307
320    li $t309, 10
321    add $t310, $t308, $t309
322    li $t311, 10
323    add $t312, $t310, $t311
324    li $t313, 10
325    add $t314, $t312, $t313
326    li $t315, 10
327    add $t316, $t314, $t315
328    li $t317, 10
329    add $t318, $t316, $t317
330    li $t319, 10
331    add $t320, $t318, $t319
332    li $t321, 10
333    add $t322, $t320, $t321
334    li $t323, 10
335    add $t324, $t322, $t323
336    li $t325, 10
337    add $t326, $t324, $t325
338    li $t327, 10
339    add $t328, $t326, $t327
340    li $t329, 10
341    add $t330, $t328, $t329
342    li $t331, 10
343    add $t332, $t330, $t331
344    li $t333, 10
345    add $t334, $t332, $t333
346    li $t335, 10
347    add $t336, $t334, $t335
348    li $t337, 10
349    add $t338, $t336, $t337
350    li $t339, 10
351    add $t340, $t338, $t339
352    li $t341, 10
353    add $t342, $t340, $t341
354    li $t343, 10
355    add $t344, $t342, $t343
356    li $t345, 10
357    add $t346, $t344, $t345
358    li $t347, 10
359    add $t348, $t346, $t347
360    li $t349, 10
361    add $t350, $t348, $t349
362    li $t351, 10
363    add $t352, $t350, $t351
364    li $t353, 10
365    add $t354, $t352, $t353
366    li $t355, 10
367    add $t356, $t354, $t355
368    li $t357, 10
369    add $t358, $t356, $t357
370    li $t359, 10
371    add $t360, $t358, $t359
372    li $t361, 10
373    add $t362, $t360, $t361
374    li $t363, 10
375    add $t364, $t362, $t363
376    li $t365, 10
377    add $t366, $t364, $t365
378    li $t367, 10
379    add $t368, $t366, $t367
380    li $t369, 10
381    add $t370, $t368, $t369
382    li $t371, 10
383    add $t372, $t370, $t371
384    li $t373, 10
385    add $t374, $t372, $t373
386    li $t375, 10
387    add $t376, $t374, $t375
388    li $t377, 10
389    add $t378, $t376, $t377
390    li $t379, 10
391    add $t380, $t378, $t379
392    li $t381, 10
393    add $t382, $t380, $t381
394    li $t383, 10
395    add $t384, $t382, $t383
396    li $t385, 10
397    add $t386, $t384, $t385
398    li $t387, 10
399    add $t388, $t386, $t387
400    li $t389, 10
401    add $t390, $t388, $t389
402    li $t391, 10
403    add $t392, $t390, $t391
404    li $t393, 10
405    add $t394, $t392, $t393
406    li $t395, 10
407    add $t396, $t394, $t395
408    li $t397, 10
409    add $t398, $t396, $t397
410    li $t399, 10
411    add $t400, $t398, $t399
412    li $t401, 10
413    add $t402, $t400, $t401
414    li $t403, 10
415    add $t404, $t402, $t403
416    li $t405, 10
417    add $t406, $t404, $t405
418    li $t407, 10
419    add $t408, $t406, $t407
420    li $t409, 10
421    add $t410, $t408, $t409
422    li $t411, 10
423    add $t412, $t410, $t411
424    li $t413, 10
425    add $t414, $t412, $t413
426    li $t415, 10
427    add $t416, $t414, $t415
428    li $t417, 10
429    add $t418, $t416, $t417
430    li $t419, 10
431    add $t420, $t418, $t419
432    li $t421, 10
433    add $t422, $t420, $t421
434    li $t423, 10
435    add $t424, $t422, $t423
436    li $t425, 10
437    add $t426, $t424, $t425
438    li $t427, 10
439    add $t428, $t426, $t427
440    li $t429, 10
441    add $t430, $t428, $t429
442    li $t431, 10
443    add $t432, $t430, $t431
444    li $t433, 10
445    add $t434, $t432, $t433
446    li $t435, 10
447    add $t436, $t434, $t435
448    li $t437, 10
449    add $t438, $t436, $t437
450    li $t439, 10
451    add $t440, $t438, $t439
452    li $t441, 10
453    add $t442, $t440, $t441
454    li $t443, 10
455    add $t444, $t442, $t443
456    li $t445, 10
457    add $t446, $t444, $t445
458    li $t447, 10
459    add $t448, $t446, $t447
460    li $t449, 10
461    add $t450, $t448, $t449
462    li $t451, 10
463    add $t452, $t450, $t451
464    li $t453, 10
465    add $t454, $t452, $t453
466    li $t455, 10
467    add $t456, $t454, $t455
468    li $t457, 10
469    add $t458, $t456, $t457
470    li $t459, 10
471    add $t460, $t458, $t459
472    li $t461, 10
473    add $t462, $t460, $t461
474    li $t463, 10
475    add $t464, $t462, $t463
476    li $t465, 10
477    add $t466, $t464, $t465
478    li $t467, 10
479    add $t468, $t466, $t467
480    li $t469, 10
481    add $t470, $t468, $t469
482    li $t471, 10
483    add $t472, $t470, $t471
484    li $t473, 10
485    add $t474, $t472, $t473
486    li $t475, 10
487    add $t476, $t474, $t475
488    li $t477, 10
489    add $t478, $t476, $t477
490    li $t479, 10
491    add $t480, $t478, $t479
492    li $t481, 10
493    add $t482, $t480, $t481
494    li $t483, 10
495    add $t484, $t482, $t483
496    li $t485, 10
497    add $t486, $t484, $t485
498    li $t487, 10
499    add $t488, $t486, $t487
500    li $t489, 10
501    add $t490, $t488, $t489
502    li $t491, 10
503    add $t492, $t490, $t491
504    li $t493, 10
505    add $t494, $t492, $t493
506    li $t495, 10
507    add $t496, $t494, $t495
508    li $t497, 10
509    add $t498, $t496, $t497
510    li $t499, 10
511    add $t500, $t498, $t499
512    li $t501, 10
513    add $t502, $t500, $t501
514    li $t503, 10
515    add $t504, $t502, $t503
516    li $t505, 10
517    add $t506, $t504, $t505
518    li $t507, 10
519    add $t508, $t506, $t507
520    li $t509, 10
521    add $t510, $t508, $t509
522    li $t511, 10
523    add $t512, $t510, $t511
524    li $t513, 10
525    add $t514, $t512, $t513
526    li $t515, 10
527    add $t516, $t514, $t515
528    li $t517, 10
529    add $t518, $t516, $t517
530    li $t519, 10
531    add $t520, $t518, $t519
532    li $t521, 10
533    add $t522, $t520, $t521
534    li $t523, 10
535    add $t524, $t522, $t523
536    li $t525, 10
537    add $t526, $t524, $t525
538    li $t527, 10
539    add $t528, $t526, $t527
540    li $t529, 10
541    add $t530, $t528, $t529
542    li $t531, 10
543    add $t532, $t530, $t531
544    li $t533, 10
545    add $t534, $t532, $t533
546    li $t535, 10
547    add $t536, $t534, $t535
548    li $t537, 10
549    add $t538, $t536, $t537
550    li $t539, 10
551    add $t540, $t538, $t539
552    li $t541, 10
553    add $t542, $t540, $t541
554    li $t543, 10
555    add $t544, $t542, $t543
556    li $t545, 10
557    add $t546, $t544, $t545
558    li $t547, 10
559    add $t548, $t546, $t547
560    li $t549, 10
561    add $t550, $t548, $t549
562    li $t551, 10
563    add $t552, $t550, $t551
564    li $t553, 10
565    add $t554, $t552, $t553
566    li $t555, 10
567    add $t556, $t554, $t555
568    li $t557, 10
569    add $t558, $t556, $t557
570    li $t559, 10
571    add $t560, $t558, $t559
572    li $t561, 10
573    add $t562, $t560, $t561
574    li $t563, 10
575    add $t564, $t562, $t563
576    li $t565, 10
577    add $t566, $t564, $t565
578    li $t567, 10
579    add $t568, $t566, $t567
580    li $t569, 10
581    add $t570, $t568, $t569
582    li $t571, 10
583    add $t572, $t570, $t571
584    li $t573, 10
585    add $t574, $t572, $t573
586    li $t575, 10
587    add $t576, $t574, $t575
588    li $t577, 10
589    add $t578, $t576, $t577
590    li $t579, 10
591    add $t580, $t578, $t579
592    li $t581, 10
593    add $t582, $t580, $t581
594    li $t583, 10
595    add $t584, $t582, $t583
596    li $t585, 10
597    add $t586, $t584, $t585
598    li $t587, 10
599    add $t588, $t586, $t587
600    li $t589, 10
601    add $t590, $t588, $t589
602    li $t591, 10
603    add $t592, $t590, $t591
604    li $t593, 10
605    add $t594, $t592, $t593
606    li $t595, 10
607    add $t596, $t594, $t595
608    li $t597, 10
609    add $t598, $t596, $t597
610    li $t599, 10
611    add $t600, $t598, $t599
612    li $t601, 10
613    add $t602, $t600, $t601
614    li $t603, 10
615    add $t604, $t602, $t603
616    li $t605, 10
617    add $t606, $t604, $t605
618    li $t607, 10
619    add $t608, $t606, $t607
620    li $t609, 10
621    add $t610, $t608, $t609
622    li $t611, 10
623    add $t612, $t610, $t611
624    li $t613, 10
625    add $t614, $t612, $t613
626    li $t615, 10
627    add $t616, $t614, $t615
628    li $t617, 10
629    add $t618, $t616, $t617
630    li $t619, 10
631    add $t620, $t618, $t619
632    li $t621, 10
633    add $t622, $t620, $t621
634    li $t623, 10
635    add $t624, $t622, $t623
636    li $t625, 10
637    add $t626, $t624, $t625
638    li $t627, 10
639    add $t628, $t626, $t627
640    li $t629, 10
641    add $t630, $t628, $t629
642    li $t631, 10
643    add $t632, $t630, $t631
644    li $t633, 10
645    add $t634, $t632, $t633
646    li $t635, 10
647    add $t636, $t634, $t635
648    li $t637, 10
649    add $t638, $t636, $t637
650    li $t639, 10
651    add $t640, $t638, $t639
652    li $t641, 10
653    add $t642, $t640, $t641
654    li $t643, 10
655    add $t644, $t642, $t643
656    li $t645, 10
657    add $t646, $t644, $t645
658    li $t647, 10
659    add $t648, $t646, $t647
660    li $t649, 10
661    add $t650, $t648, $t649
662    li $t651, 10
663    add $t652, $t650, $t651
664    li $t653, 10
665    add $t654, $t652, $t653
666    li $t
```

# MIPS Commands

The screenshot shows a MIPS assembly debugger interface. The top menu bar includes 'File', 'Edit', 'Get', 'Show/Hide Demo', 'User Guide | Unit Tests | Docs', 'Addition', 'Subtraction', 'Multiplication', 'Division', 'Hello World', 'Code Gen', 'Save String', 'Interactive', 'Decimal Decimal', 'Decimal Binary', and 'Debug'. The main window displays assembly code:

```
1 # Shows 'Hello world' at the top of the stack
2 .text
3 .globl _start
4 .data
5 _start: .asciz "Hello world\n"
6 .text
7 _start: li $t0, 111901
8 sb $t0, 111901
9 li $t1, 111902
10 sb $t1, 111902
11 li $t2, 111903
12 sb $t2, 111903
13 li $t3, 111904
14 sb $t3, 111904
15 li $t4, 111905
16 sb $t4, 111905
17 li $t5, 111906
18 sb $t5, 111906
19 li $t6, 111907
20 sb $t6, 111907
21 li $t7, 111908
22 sb $t7, 111908
23 li $t8, 111909
24 sb $t8, 111909
25 li $t9, 111910
26 sb $t9, 111910
27 li $t10, 111911
28 sb $t10, 111911
29 li $t11, 111912
30 sb $t11, 111912
31 li $t12, 111913
32 sb $t12, 111913
33 li $t13, 111914
34 sb $t13, 111914
35 li $t14, 111915
36 sb $t14, 111915
37 li $t15, 111916
38 sb $t15, 111916
39 li $t16, 111917
40 sb $t16, 111917
41 li $t17, 111918
42 sb $t17, 111918
43 li $t18, 111919
44 sb $t18, 111919
45 li $t19, 111920
46 sb $t19, 111920
47 li $t20, 111921
48 sb $t20, 111921
49 li $t21, 111922
50 sb $t21, 111922
51 li $t22, 111923
52 sb $t22, 111923
53 li $t23, 111924
54 sb $t23, 111924
55 li $t24, 111925
56 sb $t24, 111925
57 li $t25, 111926
58 sb $t25, 111926
59 li $t26, 111927
60 sb $t26, 111927
61 li $t27, 111928
62 sb $t27, 111928
63 li $t28, 111929
64 sb $t28, 111929
65 li $t29, 111930
66 sb $t29, 111930
67 li $t30, 111931
68 sb $t30, 111931
69 li $t31, 111932
70 sb $t31, 111932
71 li $t32, 111933
72 sb $t32, 111933
73 li $t33, 111934
74 sb $t33, 111934
75 li $t34, 111935
76 sb $t34, 111935
77 li $t35, 111936
78 sb $t35, 111936
79 li $t36, 111937
80 sb $t36, 111937
81 li $t37, 111938
82 sb $t37, 111938
83 li $t38, 111939
84 sb $t38, 111939
85 li $t39, 111940
86 sb $t39, 111940
87 li $t40, 111941
88 sb $t40, 111941
89 li $t41, 111942
90 sb $t41, 111942
91 li $t42, 111943
92 sb $t42, 111943
93 li $t43, 111944
94 sb $t43, 111944
95 li $t44, 111945
96 sb $t44, 111945
97 li $t45, 111946
98 sb $t45, 111946
99 li $t46, 111947
100 sb $t46, 111947
101 li $t47, 111948
102 sb $t47, 111948
103 li $t48, 111949
104 sb $t48, 111949
105 li $t49, 111950
106 sb $t49, 111950
107 li $t50, 111951
108 sb $t50, 111951
109 li $t51, 111952
110 sb $t51, 111952
111 li $t52, 111953
112 sb $t52, 111953
113 li $t53, 111954
114 sb $t53, 111954
115 li $t54, 111955
116 sb $t54, 111955
117 li $t55, 111956
118 sb $t55, 111956
119 li $t56, 111957
120 sb $t56, 111957
121 li $t57, 111958
122 sb $t57, 111958
123 li $t58, 111959
124 sb $t58, 111959
125 li $t59, 111960
126 sb $t59, 111960
127 li $t60, 111961
128 sb $t60, 111961
129 li $t61, 111962
130 sb $t61, 111962
131 li $t62, 111963
132 sb $t62, 111963
133 li $t63, 111964
134 sb $t63, 111964
135 li $t64, 111965
136 sb $t64, 111965
137 li $t65, 111966
138 sb $t65, 111966
139 li $t66, 111967
140 sb $t66, 111967
141 li $t67, 111968
142 sb $t67, 111968
143 li $t68, 111969
144 sb $t68, 111969
145 li $t69, 111970
146 sb $t69, 111970
147 li $t70, 111971
148 sb $t70, 111971
149 li $t71, 111972
150 sb $t71, 111972
151 li $t72, 111973
152 sb $t72, 111973
153 li $t73, 111974
154 sb $t73, 111974
155 li $t74, 111975
156 sb $t74, 111975
157 li $t75, 111976
158 sb $t75, 111976
159 li $t76, 111977
160 sb $t76, 111977
161 li $t77, 111978
162 sb $t77, 111978
163 li $t78, 111979
164 sb $t78, 111979
165 li $t79, 111980
166 sb $t79, 111980
167 li $t80, 111981
168 sb $t80, 111981
169 li $t81, 111982
170 sb $t81, 111982
171 li $t82, 111983
172 sb $t82, 111983
173 li $t83, 111984
174 sb $t83, 111984
175 li $t84, 111985
176 sb $t84, 111985
177 li $t85, 111986
178 sb $t85, 111986
179 li $t86, 111987
180 sb $t86, 111987
181 li $t87, 111988
182 sb $t87, 111988
183 li $t88, 111989
184 sb $t88, 111989
185 li $t89, 111990
186 sb $t89, 111990
187 li $t90, 111991
188 sb $t90, 111991
189 li $t91, 111992
190 sb $t91, 111992
191 li $t92, 111993
192 sb $t92, 111993
193 li $t93, 111994
194 sb $t93, 111994
195 li $t94, 111995
196 sb $t94, 111995
197 li $t95, 111996
198 sb $t95, 111996
199 li $t96, 111997
200 sb $t96, 111997
201 li $t97, 111998
202 sb $t97, 111998
203 li $t98, 111999
204 sb $t98, 111999
205 li $t99, 111900
206 sb $t99, 111900
207 li $t100, 111901
208 sb $t100, 111901
209 li $t101, 111902
210 sb $t101, 111902
211 li $t102, 111903
212 sb $t102, 111903
213 li $t103, 111904
214 sb $t103, 111904
215 li $t104, 111905
216 sb $t104, 111905
217 li $t105, 111906
218 sb $t105, 111906
219 li $t106, 111907
220 sb $t106, 111907
221 li $t107, 111908
222 sb $t107, 111908
223 li $t108, 111909
224 sb $t108, 111909
225 li $t109, 111910
226 sb $t109, 111910
227 li $t110, 111911
228 sb $t110, 111911
229 li $t111, 111912
230 sb $t111, 111912
231 li $t112, 111913
232 sb $t112, 111913
233 li $t113, 111914
234 sb $t113, 111914
235 li $t114, 111915
236 sb $t114, 111915
237 li $t115, 111916
238 sb $t115, 111916
239 li $t116, 111917
240 sb $t116, 111917
241 li $t117, 111918
242 sb $t117, 111918
243 li $t118, 111919
244 sb $t118, 111919
245 li $t119, 111920
246 sb $t119, 111920
247 li $t120, 111921
248 sb $t120, 111921
249 li $t121, 111922
250 sb $t121, 111922
251 li $t122, 111923
252 sb $t122, 111923
253 li $t123, 111924
254 sb $t123, 111924
255 li $t124, 111925
256 sb $t124, 111925
257 li $t125, 111926
258 sb $t125, 111926
259 li $t126, 111927
260 sb $t126, 111927
261 li $t127, 111928
262 sb $t127, 111928
263 li $t128, 111929
264 sb $t128, 111929
265 li $t129, 111930
266 sb $t129, 111930
267 li $t130, 111931
268 sb $t130, 111931
269 li $t131, 111932
270 sb $t131, 111932
271 li $t132, 111933
272 sb $t132, 111933
273 li $t133, 111934
274 sb $t133, 111934
275 li $t134, 111935
276 sb $t134, 111935
277 li $t135, 111936
278 sb $t135, 111936
279 li $t136, 111937
280 sb $t136, 111937
281 li $t137, 111938
282 sb $t137, 111938
283 li $t138, 111939
284 sb $t138, 111939
285 li $t139, 111940
286 sb $t139, 111940
287 li $t140, 111941
288 sb $t140, 111941
289 li $t141, 111942
290 sb $t141, 111942
291 li $t142, 111943
292 sb $t142, 111943
293 li $t143, 111944
294 sb $t143, 111944
295 li $t144, 111945
296 sb $t144, 111945
297 li $t145, 111946
298 sb $t145, 111946
299 li $t146, 111947
300 sb $t146, 111947
301 li $t147, 111948
302 sb $t147, 111948
303 li $t148, 111949
304 sb $t148, 111949
305 li $t149, 111950
306 sb $t149, 111950
307 li $t150, 111951
308 sb $t150, 111951
309 li $t151, 111952
310 sb $t151, 111952
311 li $t152, 111953
312 sb $t152, 111953
313 li $t153, 111954
314 sb $t153, 111954
315 li $t154, 111955
316 sb $t154, 111955
317 li $t155, 111956
318 sb $t155, 111956
319 li $t156, 111957
320 sb $t156, 111957
321 li $t157, 111958
322 sb $t157, 111958
323 li $t158, 111959
324 sb $t158, 111959
325 li $t159, 111960
326 sb $t159, 111960
327 li $t160, 111961
328 sb $t160, 111961
329 li $t161, 111962
330 sb $t161, 111962
331 li $t162, 111963
332 sb $t162, 111963
333 li $t163, 111964
334 sb $t163, 111964
335 li $t164, 111965
336 sb $t164, 111965
337 li $t165, 111966
338 sb $t165, 111966
339 li $t166, 111967
340 sb $t166, 111967
341 li $t167, 111968
342 sb $t167, 111968
343 li $t168, 111969
344 sb $t168, 111969
345 li $t169, 111970
346 sb $t169, 111970
347 li $t170, 111971
348 sb $t170, 111971
349 li $t171, 111972
350 sb $t171, 111972
351 li $t172, 111973
352 sb $t172, 111973
353 li $t173, 111974
354 sb $t173, 111974
355 li $t174, 111975
356 sb $t174, 111975
357 li $t175, 111976
358 sb $t175, 111976
359 li $t176, 111977
360 sb $t176, 111977
361 li $t177, 111978
362 sb $t177, 111978
363 li $t178, 111979
364 sb $t178, 111979
365 li $t179, 111980
366 sb $t179, 111980
367 li $t180, 111981
368 sb $t180, 111981
369 li $t181, 111982
370 sb $t181, 111982
371 li $t182, 111983
372 sb $t182, 111983
373 li $t183, 111984
374 sb $t183, 111984
375 li $t184, 111985
376 sb $t184, 111985
377 li $t185, 111986
378 sb $t185, 111986
379 li $t186, 111987
380 sb $t186, 111987
381 li $t187, 111988
382 sb $t187, 111988
383 li $t188, 111989
384 sb $t188, 111989
385 li $t189, 111990
386 sb $t189, 111990
387 li $t190, 111991
388 sb $t190, 111991
389 li $t191, 111992
390 sb $t191, 111992
391 li $t192, 111993
392 sb $t192, 111993
393 li $t193, 111994
394 sb $t193, 111994
395 li $t194, 111995
396 sb $t194, 111995
397 li $t195, 111996
398 sb $t195, 111996
399 li $t196, 111997
400 sb $t196, 111997
401 li $t197, 111998
402 sb $t197, 111998
403 li $t198, 111999
404 sb $t198, 111999
405 li $t199, 111900
406 sb $t199, 111900
407 li $t200, 111901
408 sb $t200, 111901
409 li $t201, 111902
410 sb $t201, 111902
411 li $t202, 111903
412 sb $t202, 111903
413 li $t203, 111904
414 sb $t203, 111904
415 li $t204, 111905
416 sb $t204, 111905
417 li $t205, 111906
418 sb $t205, 111906
419 li $t206, 111907
420 sb $t206, 111907
421 li $t207, 111908
422 sb $t207, 111908
423 li $t208, 111909
424 sb $t208, 111909
425 li $t209, 111910
426 sb $t209, 111910
427 li $t210, 111911
428 sb $t210, 111911
429 li $t211, 111912
430 sb $t211, 111912
431 li $t212, 111913
432 sb $t212, 111913
433 li $t213, 111914
434 sb $t213, 111914
435 li $t214, 111915
436 sb $t214, 111915
437 li $t215, 111916
438 sb $t215, 111916
439 li $t216, 111917
440 sb $t216, 111917
441 li $t217, 111918
442 sb $t217, 111918
443 li $t218, 111919
444 sb $t218, 111919
445 li $t219, 111920
446 sb $t219, 111920
447 li $t220, 111921
448 sb $t220, 111921
449 li $t221, 111922
450 sb $t221, 111922
451 li $t222, 111923
452 sb $t222, 111923
453 li $t223, 111924
454 sb $t223, 111924
455 li $t224, 111925
456 sb $t224, 111925
457 li $t225, 111926
458 sb $t225, 111926
459 li $t226, 111927
460 sb $t226, 111927
461 li $t227, 111928
462 sb $t227, 111928
463 li $t228, 111929
464 sb $t228, 111929
465 li $t229, 111930
466 sb $t229, 111930
467 li $t230, 111931
468 sb $t230, 111931
469 li $t231, 111932
470 sb $t231, 111932
471 li $t232, 111933
472 sb $t232, 111933
473 li $t233, 111934
474 sb $t233, 111934
475 li $t234, 111935
476 sb $t234, 111935
477 li $t235, 111936
478 sb $t235, 111936
479 li $t236, 111937
480 sb $t236, 111937
481 li $t237, 111938
482 sb $t237, 111938
483 li $t238, 111939
484 sb $t238, 111939
485 li $t239, 111940
486 sb $t239, 111940
487 li $t240, 111941
488 sb $t240, 111941
489 li $t241, 111942
490 sb $t241, 111942
491 li $t242, 111943
492 sb $t242, 111943
493 li $t243, 111944
494 sb $t243, 111944
495 li $t244, 111945
496 sb $t244, 111945
497 li $t245, 111946
498 sb $t245, 111946
499 li $t246, 111947
500 sb $t246, 111947
501 li $t247, 111948
502 sb $t247, 111948
503 li $t248, 111949
504 sb $t248, 111949
505 li $t249, 111950
506 sb $t249, 111950
507 li $t250, 111951
508 sb $t250, 111951
509 li $t251, 111952
510 sb $t251, 111952
511 li $t252, 111953
512 sb $t252, 111953
513 li $t253, 111954
514 sb $t253, 111954
515 li $t254, 111955
516 sb $t254, 111955
517 li $t255, 111956
518 sb $t255, 111956
519 li $t256, 111957
520 sb $t256, 111957
521 li $t257, 111958
522 sb $t257, 111958
523 li $t258, 111959
524 sb $t258, 111959
525 li $t259, 111960
526 sb $t259, 111960
527 li $t260, 111961
528 sb $t260, 111961
529 li $t261, 111962
530 sb $t261, 111962
531 li $t262, 111963
532 sb $t262, 111963
533 li $t263, 111964
534 sb $t263, 111964
535 li $t264, 111965
536 sb $t264, 111965
537 li $t265, 111966
538 sb $t265, 111966
539 li $t266, 111967
540 sb $t266, 111967
541 li $t267, 111968
542 sb $t267, 111968
543 li $t268, 111969
544 sb $t268, 111969
545 li $t269, 111970
546 sb $t269, 111970
547 li $t270, 111971
548 sb $t270, 111971
549 li $t271, 111972
550 sb $t271, 111972
551 li $t272, 111973
552 sb $t272, 111973
553 li $t273, 111974
554 sb $t273, 111974
555 li $t274, 111975
556 sb $t274, 111975
557 li $t275, 111976
558 sb $t275, 111976
559 li $t276, 111977
560 sb $t276, 111977
561 li $t277, 111978
562 sb $t277, 111978
563 li $t278, 111979
564 sb $t278, 111979
565 li $t279, 111980
566 sb $t279, 111980
567 li $t280, 111981
568 sb $t280, 111981
569 li $t281, 111982
570 sb $t281, 111982
571 li $t282, 111983
572 sb $t282, 111983
573 li $t283, 111984
574 sb $t283, 111984
575 li $t284, 111985
576 sb $t284, 111985
577 li $t285, 111986
578 sb $t285, 111986
579 li $t286, 111987
580 sb $t286, 111987
581 li $t287, 111988
582 sb $t287, 111988
583 li $t288, 111989
584 sb $t288, 111989
585 li $t289, 111990
586 sb $t289, 111990
587 li $t290, 111991
588 sb $t290, 111991
589 li $t291, 111992
590 sb $t291, 111992
591 li $t292, 111993
592 sb $t292, 111993
593 li $t293, 111994
594 sb $t293, 111994
595 li $t294, 111995
596 sb $t294, 111995
597 li $t295, 111996
598 sb $t295, 111996
599 li $t296, 111997
600 sb $t296, 111997
601 li $t297, 111998
602 sb $t297, 111998
603 li $t298, 111999
604 sb $t298, 111999
605 li $t299, 111900
606 sb $t299, 111900
607 li $t300, 111901
608 sb $t300, 111901
609 li $t301, 111902
610 sb $t301, 111902
611 li $t302, 111903
612 sb $t302, 111903
613 li $t303, 111904
614 sb $t303, 111904
615 li $t304, 111905
616 sb $t304, 111905
617 li $t305, 111906
618 sb $t305, 111906
619 li $t306, 111907
620 sb $t306, 111907
621 li $t307, 111908
622 sb $t307, 111908
623 li $t308, 111909
624 sb $t308, 111909
625 li $t309, 111910
626 sb $t309, 111910
627 li $t310, 111911
628 sb $t310, 111911
629 li $t311, 111912
630 sb $t311, 111912
631 li $t312, 111913
632 sb $t312, 111913
633 li $t313, 111914
634 sb $t313, 111914
635 li $t314, 111915
636 sb $t314, 111915
637 li $t315, 111916
638 sb $t315, 111916
639 li $t316, 111917
640 sb $t316, 111917
641 li $t317, 111918
642 sb $t317, 111918
643 li $t318, 111919
644 sb $t318, 111919
645 li $t319, 111920
646 sb $t319, 111920
647 li $t320, 111921
648 sb $t320, 111921
649 li $t321, 111922
650 sb $t321, 111922
651 li $t322, 111923
652 sb $t322, 111923
653 li $t323, 111924
654 sb $t323, 111924
655 li $t324, 111925
656 sb $t324, 111925
657 li $t325, 111926
658 sb $t325, 111926
659 li $t326, 111927
660 sb $t326, 111927
661 li $t327, 111928
662 sb $t327, 111928
663 li $t328, 111929
664 sb $t328, 111929
665 li $t329, 111930
666 sb $t329, 111930
667 li $t330, 111931
668 sb $t330, 111931
669 li $t331, 111932
670 sb $t331, 111932
671 li $t332, 111933
672 sb $t332, 111933
673 li $t333, 111934
674 sb $t333, 111934
675 li $t334, 111935
676 sb $t334, 111935
677 li $t335, 111936
678 sb $t335, 111936
679 li $t336, 111937
680 sb $t336, 111937
681 li $t337, 111938
682 sb $t337, 111938
683 li $t338, 111939
684 sb $t338, 111939
685 li $t339, 111940
686 sb $t339, 111940
687 li $t340, 111941
688 sb $t340, 111941
689 li $t341, 111942
690 sb $t341, 111942
691 li $t342, 111943
692 sb $t342, 111943
693 li $t343, 111944
694 sb $t343, 111944
695 li $t344, 111945
696 sb $t344, 111945
697 li $t345, 111946
698 sb $t345, 111946
699 li $t346, 111947
700 sb $t3
```

# MIPS Commands

The screenshot shows a MIPS assembly editor interface. At the top, there's a menu bar with 'File', 'Edit', 'Get', 'Show/Hide Demo', 'Addition', 'Subtraction', 'Multiplication', 'Division', 'Hello World', 'Code Gen', 'Save String', 'Interactive', 'Decimal Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with 'Step', 'Run', 'Break', 'Create auto stepping', and 'Stop' buttons. A status bar at the bottom right says 'User Guide | Unit Tests | Docs'. The main area has tabs for 'S' (selected), 'T', 'A', 'V', 'Stack', and 'Log'. The assembly code in the 'S' tab is:

```
1 # Shows 'Hello world' at the top of the stack
2 .text
3 .globl _start
4 .data
5 _start: .asciiz "Hello world\n"
6 .text
7 _start: li $t0, _start
8 sb $t0, 11($sp)
9 li $t1, 11($sp)
10 sb $t1, 11($sp)
11 li $t2, 11($sp)
12 sb $t2, 11($sp)
13 addi $t3, $t0, $t2
14 addi $t4, $t1, $t2
15 addi $t5, $t3, $t4
16 addi $t6, $t5, $t2
17 addi $t7, $t6, $t2
18 addi $t8, $t7, $t2
19 addi $t9, $t8, $t2
20 addi $t10, $t9, $t2
21 addi $t11, $t10, $t2
22 addi $t12, $t11, $t2
23 addi $t13, $t12, $t2
24 addi $t14, $t13, $t2
25 addi $t15, $t14, $t2
26 addi $t16, $t15, $t2
27 addi $t17, $t16, $t2
28 addi $t18, $t17, $t2
29 addi $t19, $t18, $t2
30 addi $t20, $t19, $t2
31 addi $t21, $t20, $t2
32 addi $t22, $t21, $t2
33 addi $t23, $t22, $t2
34 addi $t24, $t23, $t2
35 addi $t25, $t24, $t2
36 addi $t26, $t25, $t2
37 addi $t27, $t26, $t2
38 addi $t28, $t27, $t2
39 addi $t29, $t28, $t2
40 addi $t30, $t29, $t2
41 addi $t31, $t30, $t2
42 addi $t32, $t31, $t2
43 addi $t33, $t32, $t2
44 addi $t34, $t33, $t2
45 addi $t35, $t34, $t2
46 addi $t36, $t35, $t2
47 addi $t37, $t36, $t2
48 addi $t38, $t37, $t2
49 addi $t39, $t38, $t2
50 addi $t40, $t39, $t2
51 addi $t41, $t40, $t2
52 addi $t42, $t41, $t2
53 addi $t43, $t42, $t2
54 addi $t44, $t43, $t2
55 addi $t45, $t44, $t2
56 addi $t46, $t45, $t2
57 addi $t47, $t46, $t2
58 addi $t48, $t47, $t2
59 addi $t49, $t48, $t2
60 addi $t50, $t49, $t2
61 addi $t51, $t50, $t2
62 addi $t52, $t51, $t2
63 addi $t53, $t52, $t2
64 addi $t54, $t53, $t2
65 addi $t55, $t54, $t2
66 addi $t56, $t55, $t2
67 addi $t57, $t56, $t2
68 addi $t58, $t57, $t2
69 addi $t59, $t58, $t2
70 addi $t60, $t59, $t2
71 addi $t61, $t60, $t2
72 addi $t62, $t61, $t2
73 addi $t63, $t62, $t2
74 addi $t64, $t63, $t2
75 addi $t65, $t64, $t2
76 addi $t66, $t65, $t2
77 addi $t67, $t66, $t2
78 addi $t68, $t67, $t2
79 addi $t69, $t68, $t2
80 addi $t70, $t69, $t2
81 addi $t71, $t70, $t2
82 addi $t72, $t71, $t2
83 addi $t73, $t72, $t2
84 addi $t74, $t73, $t2
85 addi $t75, $t74, $t2
86 addi $t76, $t75, $t2
87 addi $t77, $t76, $t2
88 addi $t78, $t77, $t2
89 addi $t79, $t78, $t2
90 addi $t80, $t79, $t2
91 addi $t81, $t80, $t2
92 addi $t82, $t81, $t2
93 addi $t83, $t82, $t2
94 addi $t84, $t83, $t2
95 addi $t85, $t84, $t2
96 addi $t86, $t85, $t2
97 addi $t87, $t86, $t2
98 addi $t88, $t87, $t2
99 addi $t89, $t88, $t2
100 addi $t90, $t89, $t2
101 addi $t91, $t90, $t2
102 addi $t92, $t91, $t2
103 addi $t93, $t92, $t2
104 addi $t94, $t93, $t2
105 addi $t95, $t94, $t2
106 addi $t96, $t95, $t2
107 addi $t97, $t96, $t2
108 addi $t98, $t97, $t2
109 addi $t99, $t98, $t2
110 addi $t100, $t99, $t2
111 addi $t101, $t100, $t2
112 addi $t102, $t101, $t2
113 addi $t103, $t102, $t2
114 addi $t104, $t103, $t2
115 addi $t105, $t104, $t2
116 addi $t106, $t105, $t2
117 addi $t107, $t106, $t2
118 addi $t108, $t107, $t2
119 addi $t109, $t108, $t2
120 addi $t110, $t109, $t2
121 addi $t111, $t110, $t2
122 addi $t112, $t111, $t2
123 addi $t113, $t112, $t2
124 addi $t114, $t113, $t2
125 addi $t115, $t114, $t2
126 addi $t116, $t115, $t2
127 addi $t117, $t116, $t2
128 addi $t118, $t117, $t2
129 addi $t119, $t118, $t2
130 addi $t120, $t119, $t2
131 addi $t121, $t120, $t2
132 addi $t122, $t121, $t2
133 addi $t123, $t122, $t2
134 addi $t124, $t123, $t2
135 addi $t125, $t124, $t2
136 addi $t126, $t125, $t2
137 addi $t127, $t126, $t2
138 addi $t128, $t127, $t2
139 addi $t129, $t128, $t2
140 addi $t130, $t129, $t2
141 addi $t131, $t130, $t2
142 addi $t132, $t131, $t2
143 addi $t133, $t132, $t2
144 addi $t134, $t133, $t2
145 addi $t135, $t134, $t2
146 addi $t136, $t135, $t2
147 addi $t137, $t136, $t2
148 addi $t138, $t137, $t2
149 addi $t139, $t138, $t2
150 addi $t140, $t139, $t2
151 addi $t141, $t140, $t2
152 addi $t142, $t141, $t2
153 addi $t143, $t142, $t2
154 addi $t144, $t143, $t2
155 addi $t145, $t144, $t2
156 addi $t146, $t145, $t2
157 addi $t147, $t146, $t2
158 addi $t148, $t147, $t2
159 addi $t149, $t148, $t2
160 addi $t150, $t149, $t2
161 addi $t151, $t150, $t2
162 addi $t152, $t151, $t2
163 addi $t153, $t152, $t2
164 addi $t154, $t153, $t2
165 addi $t155, $t154, $t2
166 addi $t156, $t155, $t2
167 addi $t157, $t156, $t2
168 addi $t158, $t157, $t2
169 addi $t159, $t158, $t2
170 addi $t160, $t159, $t2
171 addi $t161, $t160, $t2
172 addi $t162, $t161, $t2
173 addi $t163, $t162, $t2
174 addi $t164, $t163, $t2
175 addi $t165, $t164, $t2
176 addi $t166, $t165, $t2
177 addi $t167, $t166, $t2
178 addi $t168, $t167, $t2
179 addi $t169, $t168, $t2
180 addi $t170, $t169, $t2
181 addi $t171, $t170, $t2
182 addi $t172, $t171, $t2
183 addi $t173, $t172, $t2
184 addi $t174, $t173, $t2
185 addi $t175, $t174, $t2
186 addi $t176, $t175, $t2
187 addi $t177, $t176, $t2
188 addi $t178, $t177, $t2
189 addi $t179, $t178, $t2
190 addi $t180, $t179, $t2
191 addi $t181, $t180, $t2
192 addi $t182, $t181, $t2
193 addi $t183, $t182, $t2
194 addi $t184, $t183, $t2
195 addi $t185, $t184, $t2
196 addi $t186, $t185, $t2
197 addi $t187, $t186, $t2
198 addi $t188, $t187, $t2
199 addi $t189, $t188, $t2
200 addi $t190, $t189, $t2
201 addi $t191, $t190, $t2
202 addi $t192, $t191, $t2
203 addi $t193, $t192, $t2
204 addi $t194, $t193, $t2
205 addi $t195, $t194, $t2
206 addi $t196, $t195, $t2
207 addi $t197, $t196, $t2
208 addi $t198, $t197, $t2
209 addi $t199, $t198, $t2
210 addi $t200, $t199, $t2
211 addi $t201, $t200, $t2
212 addi $t202, $t201, $t2
213 addi $t203, $t202, $t2
214 addi $t204, $t203, $t2
215 addi $t205, $t204, $t2
216 addi $t206, $t205, $t2
217 addi $t207, $t206, $t2
218 addi $t208, $t207, $t2
219 addi $t209, $t208, $t2
220 addi $t210, $t209, $t2
221 addi $t211, $t210, $t2
222 addi $t212, $t211, $t2
223 addi $t213, $t212, $t2
224 addi $t214, $t213, $t2
225 addi $t215, $t214, $t2
226 addi $t216, $t215, $t2
227 addi $t217, $t216, $t2
228 addi $t218, $t217, $t2
229 addi $t219, $t218, $t2
230 addi $t220, $t219, $t2
231 addi $t221, $t220, $t2
232 addi $t222, $t221, $t2
233 addi $t223, $t222, $t2
234 addi $t224, $t223, $t2
235 addi $t225, $t224, $t2
236 addi $t226, $t225, $t2
237 addi $t227, $t226, $t2
238 addi $t228, $t227, $t2
239 addi $t229, $t228, $t2
240 addi $t230, $t229, $t2
241 addi $t231, $t230, $t2
242 addi $t232, $t231, $t2
243 addi $t233, $t232, $t2
244 addi $t234, $t233, $t2
245 addi $t235, $t234, $t2
246 addi $t236, $t235, $t2
247 addi $t237, $t236, $t2
248 addi $t238, $t237, $t2
249 addi $t239, $t238, $t2
250 addi $t240, $t239, $t2
251 addi $t241, $t240, $t2
252 addi $t242, $t241, $t2
253 addi $t243, $t242, $t2
254 addi $t244, $t243, $t2
255 addi $t245, $t244, $t2
256 addi $t246, $t245, $t2
257 addi $t247, $t246, $t2
258 addi $t248, $t247, $t2
259 addi $t249, $t248, $t2
260 addi $t250, $t249, $t2
261 addi $t251, $t250, $t2
262 addi $t252, $t251, $t2
263 addi $t253, $t252, $t2
264 addi $t254, $t253, $t2
265 addi $t255, $t254, $t2
266 addi $t256, $t255, $t2
267 addi $t257, $t256, $t2
268 addi $t258, $t257, $t2
269 addi $t259, $t258, $t2
270 addi $t260, $t259, $t2
271 addi $t261, $t260, $t2
272 addi $t262, $t261, $t2
273 addi $t263, $t262, $t2
274 addi $t264, $t263, $t2
275 addi $t265, $t264, $t2
276 addi $t266, $t265, $t2
277 addi $t267, $t266, $t2
278 addi $t268, $t267, $t2
279 addi $t269, $t268, $t2
280 addi $t270, $t269, $t2
281 addi $t271, $t270, $t2
282 addi $t272, $t271, $t2
283 addi $t273, $t272, $t2
284 addi $t274, $t273, $t2
285 addi $t275, $t274, $t2
286 addi $t276, $t275, $t2
287 addi $t277, $t276, $t2
288 addi $t278, $t277, $t2
289 addi $t279, $t278, $t2
290 addi $t280, $t279, $t2
291 addi $t281, $t280, $t2
292 addi $t282, $t281, $t2
293 addi $t283, $t282, $t2
294 addi $t284, $t283, $t2
295 addi $t285, $t284, $t2
296 addi $t286, $t285, $t2
297 addi $t287, $t286, $t2
298 addi $t288, $t287, $t2
299 addi $t289, $t288, $t2
300 addi $t290, $t289, $t2
301 addi $t291, $t290, $t2
302 addi $t292, $t291, $t2
303 addi $t293, $t292, $t2
304 addi $t294, $t293, $t2
305 addi $t295, $t294, $t2
306 addi $t296, $t295, $t2
307 addi $t297, $t296, $t2
308 addi $t298, $t297, $t2
309 addi $t299, $t298, $t2
310 addi $t300, $t299, $t2
311 addi $t301, $t300, $t2
312 addi $t302, $t301, $t2
313 addi $t303, $t302, $t2
314 addi $t304, $t303, $t2
315 addi $t305, $t304, $t2
316 addi $t306, $t305, $t2
317 addi $t307, $t306, $t2
318 addi $t308, $t307, $t2
319 addi $t309, $t308, $t2
320 addi $t310, $t309, $t2
321 addi $t311, $t310, $t2
322 addi $t312, $t311, $t2
323 addi $t313, $t312, $t2
324 addi $t314, $t313, $t2
325 addi $t315, $t314, $t2
326 addi $t316, $t315, $t2
327 addi $t317, $t316, $t2
328 addi $t318, $t317, $t2
329 addi $t319, $t318, $t2
330 addi $t320, $t319, $t2
331 addi $t321, $t320, $t2
332 addi $t322, $t321, $t2
333 addi $t323, $t322, $t2
334 addi $t324, $t323, $t2
335 addi $t325, $t324, $t2
336 addi $t326, $t325, $t2
337 addi $t327, $t326, $t2
338 addi $t328, $t327, $t2
339 addi $t329, $t328, $t2
340 addi $t330, $t329, $t2
341 addi $t331, $t330, $t2
342 addi $t332, $t331, $t2
343 addi $t333, $t332, $t2
344 addi $t334, $t333, $t2
345 addi $t335, $t334, $t2
346 addi $t336, $t335, $t2
347 addi $t337, $t336, $t2
348 addi $t338, $t337, $t2
349 addi $t339, $t338, $t2
350 addi $t340, $t339, $t2
351 addi $t341, $t340, $t2
352 addi $t342, $t341, $t2
353 addi $t343, $t342, $t2
354 addi $t344, $t343, $t2
355 addi $t345, $t344, $t2
356 addi $t346, $t345, $t2
357 addi $t347, $t346, $t2
358 addi $t348, $t347, $t2
359 addi $t349, $t348, $t2
360 addi $t350, $t349, $t2
361 addi $t351, $t350, $t2
362 addi $t352, $t351, $t2
363 addi $t353, $t352, $t2
364 addi $t354, $t353, $t2
365 addi $t355, $t354, $t2
366 addi $t356, $t355, $t2
367 addi $t357, $t356, $t2
368 addi $t358, $t357, $t2
369 addi $t359, $t358, $t2
370 addi $t360, $t359, $t2
371 addi $t361, $t360, $t2
372 addi $t362, $t361, $t2
373 addi $t363, $t362, $t2
374 addi $t364, $t363, $t2
375 addi $t365, $t364, $t2
376 addi $t366, $t365, $t2
377 addi $t367, $t366, $t2
378 addi $t368, $t367, $t2
379 addi $t369, $t368, $t2
380 addi $t370, $t369, $t2
381 addi $t371, $t370, $t2
382 addi $t372, $t371, $t2
383 addi $t373, $t372, $t2
384 addi $t374, $t373, $t2
385 addi $t375, $t374, $t2
386 addi $t376, $t375, $t2
387 addi $t377, $t376, $t2
388 addi $t378, $t377, $t2
389 addi $t379, $t378, $t2
390 addi $t380, $t379, $t2
391 addi $t381, $t380, $t2
392 addi $t382, $t381, $t2
393 addi $t383, $t382, $t2
394 addi $t384, $t383, $t2
395 addi $t385, $t384, $t2
396 addi $t386, $t385, $t2
397 addi $t387, $t386, $t2
398 addi $t388, $t387, $t2
399 addi $t389, $t388, $t2
400 addi $t390, $t389, $t2
401 addi $t391, $t390, $t2
402 addi $t392, $t391, $t2
403 addi $t393, $t392, $t2
404 addi $t394, $t393, $t2
405 addi $t395, $t394, $t2
406 addi $t396, $t395, $t2
407 addi $t397, $t396, $t2
408 addi $t398, $t397, $t2
409 addi $t399, $t398, $t2
410 addi $t400, $t399, $t2
411 addi $t401, $t400, $t2
412 addi $t402, $t401, $t2
413 addi $t403, $t402, $t2
414 addi $t404, $t403, $t2
415 addi $t405, $t404, $t2
416 addi $t406, $t405, $t2
417 addi $t407, $t406, $t2
418 addi $t408, $t407, $t2
419 addi $t409, $t408, $t2
420 addi $t410, $t409, $t2
421 addi $t411, $t410, $t2
422 addi $t412, $t411, $t2
423 addi $t413, $t412, $t2
424 addi $t414, $t413, $t2
425 addi $t415, $t414, $t2
426 addi $t416, $t415, $t2
427 addi $t417, $t416, $t2
428 addi $t418, $t417, $t2
429 addi $t419, $t418, $t2
430 addi $t420, $t419, $t2
431 addi $t421, $t420, $t2
432 addi $t422, $t421, $t2
433 addi $t423, $t422, $t2
434 addi $t424, $t423, $t2
435 addi $t425, $t424, $t2
436 addi $t426, $t425, $t2
437 addi $t427, $t426, $t2
438 addi $t428, $t427, $t2
439 addi $t429, $t428, $t2
440 addi $t430, $t429, $t2
441 addi $t431, $t430, $t2
442 addi $t432, $t431, $t2
443 addi $t433, $t432, $t2
444 addi $t434, $t433, $t2
445 addi $t435, $t434, $t2
446 addi $t436, $t435, $t2
447 addi $t437, $t436, $t2
448 addi $t438, $t437, $t2
449 addi $t439, $t438, $t2
450 addi $t440, $t439, $t2
451 addi $t441, $t440, $t2
452 addi $t442, $t441, $t2
453 addi $t443, $t442, $t2
454 addi $t444, $t443, $t2
455 addi $t445, $t444, $t2
456 addi $t446, $t445, $t2
457 addi $t447, $t446, $t2
458 addi $t448, $t447, $t2
459 addi $t449, $t448, $t2
460 addi $t450, $t449, $t2
461 addi $t451, $t450, $t2
462 addi $t452, $t451, $t2
463 addi $t453, $t452, $t2
464 addi $t454, $t453, $t2
465 addi $t455, $t454, $t2
466 addi $t456, $t455, $t2
467 addi $t457, $t456, $t2
468 addi $t458, $t457, $t2
469 addi $t459, $t458, $t2
470 addi $t460, $t459, $t2
471 addi $t461, $t460, $t2
472 addi $t462, $t461, $t2
473 addi $t463, $t462, $t2
474 addi $t464, $t463, $t2
475 addi $t465, $t464, $t2
476 addi $t466, $t465, $t2
477 addi $t467, $t466, $t2
478 addi $t468, $t467, $t2
479 addi $t469, $t468, $t2
480 addi $t470, $t469, $t2
481 addi $t471, $t470, $t2
482 addi $t472, $t471, $t2
483 addi $t473, $t472, $t2
484 addi $t474, $t473, $t2
485 addi $t475, $t474, $t2
486 addi $t476, $t475, $t2
487 addi $t477, $t476, $t2
488 addi $t478, $t477, $t2
489 addi $t479, $t478, $t2
490 addi $t480, $t479, $t2
491 addi $t481, $t480, $t2
492 addi $t482, $t481, $t2
493 addi $t483, $t482, $t2
494 addi $t484, $t483, $t2
495 addi $t485, $t484, $t2
496 addi $t486, $t485, $t2
497 addi $t487, $t486, $t2
498 addi $t488, $t487, $t2
499 addi $t489, $t488, $t2
500 addi $t490, $t489, $t2
501 addi $t491, $t490, $t2
502 addi $t492, $t491, $t2
503 addi $t493, $t492, $t2
504 addi $t494, $t493, $t2
505 addi $t495, $t494, $t2
506 addi $t496, $t495, $t2
507 addi $t497, $t496, $t2
508 addi $t498, $t497, $t2
509 addi $t499, $t498, $t2
510 addi $t500, $t499, $t2
511 addi $t501, $t500, $t2
512 addi $t502, $t501, $t2
513 addi $t503, $t502, $t2
514 addi $t504, $t503, $t2
515 addi $t505, $t504, $t2
516 addi $t506, $t505, $t2
517 addi $t507, $t506, $t2
518 addi $t508, $t507, $t2
519 addi $t509, $t508, $t2
520 addi $t510, $t509, $t2
521 addi $t511, $t510, $t2
522 addi $t512, $t511, $t2
523 addi $t513, $t512, $t2
524 addi $t514, $t513, $t2
525 addi $t515, $t514, $t2
526 addi $t516, $t515, $t2
527 addi $t517, $t516, $t2
528 addi $t518, $t517, $t2
529 addi $t519, $t518, $t2
530 addi $t520, $t519, $t2
531 addi $t521, $t520, $t2
532 addi $t522, $t521, $t2
533 addi $t523, $t522, $t2
534 addi $t524, $t523, $t2
535 addi $t525, $t524, $t2
536 addi $t526, $t525, $t2
537 addi $t527, $t526, $t2
538 addi $t528, $t527, $t2
539 addi $t529, $t528, $t2
540 addi $t530, $t529, $t2
541 addi $t531, $t530, $t2
542 addi $t532, $t531, $t2
543 addi $t533, $t532, $t2
544 addi $t534, $t533, $t2
545 addi $t535, $t534, $t2
546 addi $t536, $t535, $t2
547 addi $t537, $t536, $t2
548 addi $t538, $t537, $t2
549 addi $t539, $t538, $t2
550 addi $t540, $t539, $t2
551 addi $t541, $t540, $t2
552 addi $t542, $t541, $t2
553 addi $t543, $t542, $t2
554 addi $t544, $t543, $t2
555 addi $t545, $t544, $t2
556 addi $t546, $t545, $t2
557 addi $t547, $t546, $t2
558 addi $t548, $t547, $t2
559 addi $t549, $t548, $t2
560 addi $t550, $t549, $t2
561 addi $t551, $t550, $t2
562 addi $t552, $t551, $t2
563 addi $t553, $t552, $t2
564 addi $t554, $t553, $t2
565 addi $t555, $t554, $t2
566 addi $t556, $t555, $t2
567 addi $t557, $t556, $t2
568 addi $t558, $t557, $t2
569 addi $t559, $t558, $t2
570 addi $t560, $t559, $t2
571 addi $t561, $t560, $t2
572 addi $t562, $t561, $t2
573 addi $t563, $t562, $t2
574 addi $t564, $t563, $t2
575 addi $t565, $t564, $t2
576 addi $t566, $t565, $t2
577 addi $t567, $t566, $t2
578 addi $t568, $t567, $t2
579 addi $t569, $t568, $t2
580 addi $t570, $t569, $t2
581 addi $t571, $t570, $t2
582 addi $t572, $t571, $t2
583 addi $t573, $t572, $t2
584 addi $t574, $t573, $t2
585 addi $t575, $t574, $t2
586 addi $t576, $t575, $t2
587 addi $t577, $t576, $t2
588 addi $t578, $t577, $t2
589 addi $t579, $t578, $t2
590 addi $t580, $t579, $t2
591 addi $t581, $t580, $t2
592 addi $t582, $t581, $t2
593 addi $t583, $t582, $t2
594 addi $t584, $t583, $t2
595 addi $t585, $t584, $t2
596 addi $t586, $t585, $t2
597 addi $t587, $t586, $t2
598 addi $t588, $t587, $t2
599 addi $t589,
```

## MIPS Commands

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

# Challenge:

Line: 3 Go! Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

Step Run  Enable auto switching

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

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

WeMIPS

## (Demo with WeMIPS)

# Today's Topics



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

# 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:	1	Go!
-------	---	-----

Show/Hide Demos

Addition Doubler	Stav	Looper	Stack Test	Hello World	Code Gen Save String	Interactive
Binary2 Decimal	Decimal2 Binary	Debug				

[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 loop 26 times
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                 # Else, jump back to SETUP
11 DONE: ADDI $t0, $zero, 0 # Null (0) to terminate string
12 SB $t0, 0($sp)          # Add null to stack
13 ADDI $sp, $sp, -27      # Set up stack to print
14
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)

Enable auto switching

S T A V Stack Log

Emulation complete, returning to line 1

abcdefghijklmnopqrstuvwxyz

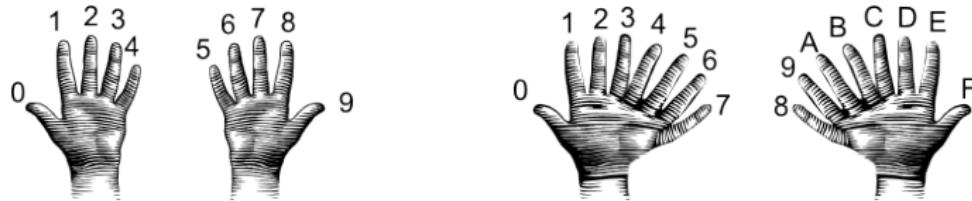


# Today's Topics



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

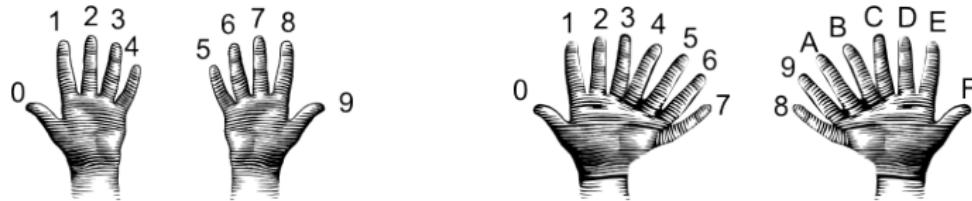
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

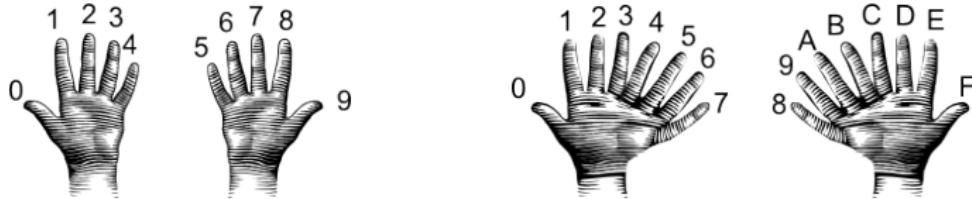
# Hexadecimal to Decimal: Converting Between Bases



(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.

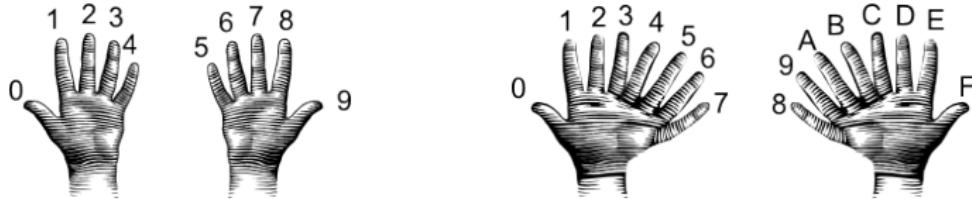
# Hexadecimal to Decimal: Converting Between Bases



(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?

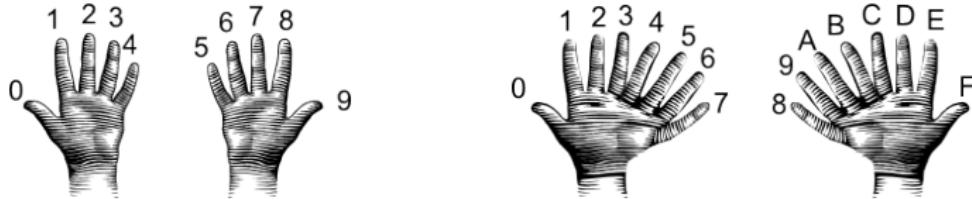
# Hexadecimal to Decimal: Converting Between Bases



(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.

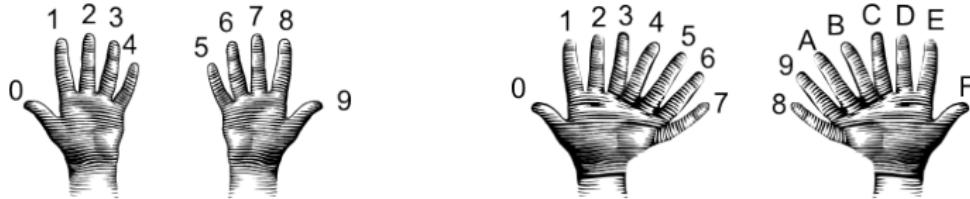
# Hexadecimal to Decimal: Converting Between Bases



(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.

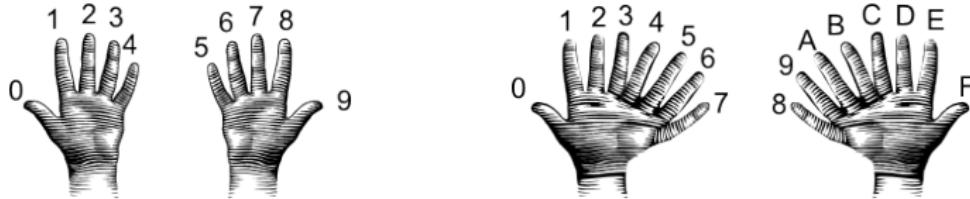
# Hexadecimal to Decimal: Converting Between Bases



(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):

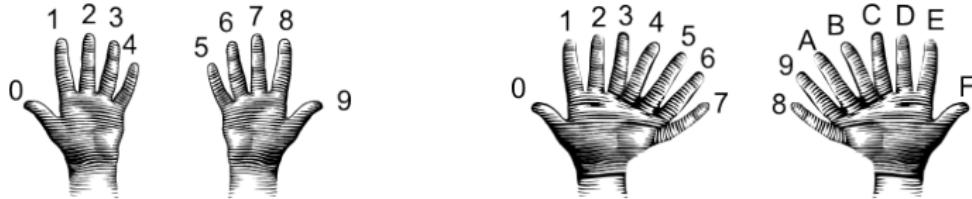
- Convert first digit to decimal and multiple by 16.
  - Convert second digit to decimal and add to total.
  - Example: what is 2A as a decimal number?

2 in decimal is 2.  $2 \times 16$  is 32.

A in decimal digits is 10.

$32 + 10$  is 42.

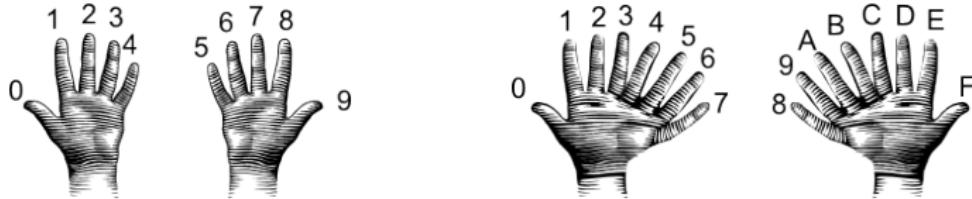
# Hexadecimal to Decimal: Converting Between Bases



(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?

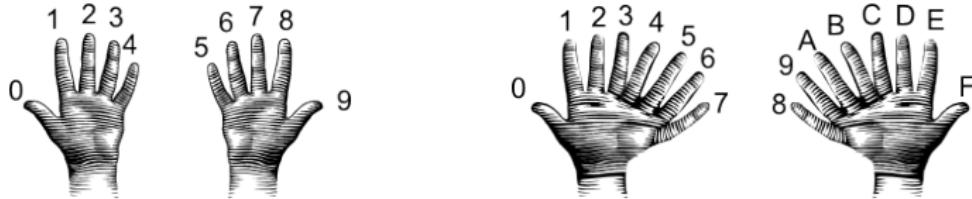
# Hexadecimal to Decimal: Converting Between Bases



(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.

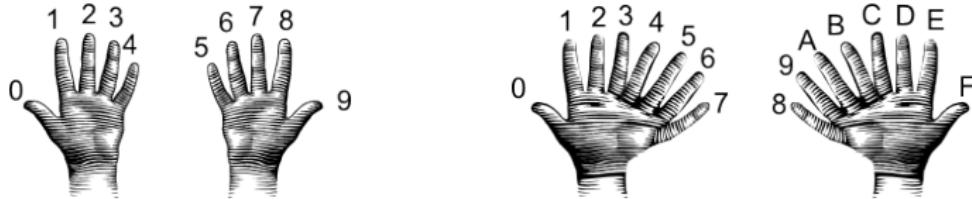
# Hexadecimal to Decimal: Converting Between Bases



(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.

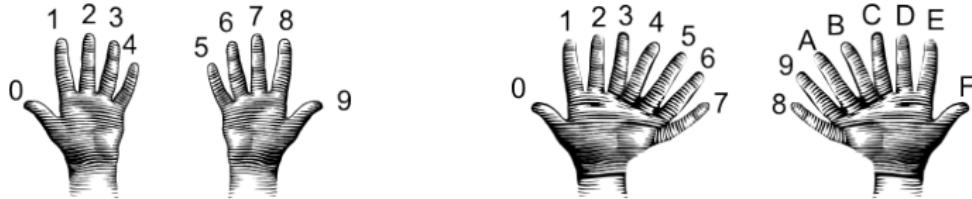
# Hexadecimal to Decimal: Converting Between Bases



(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.

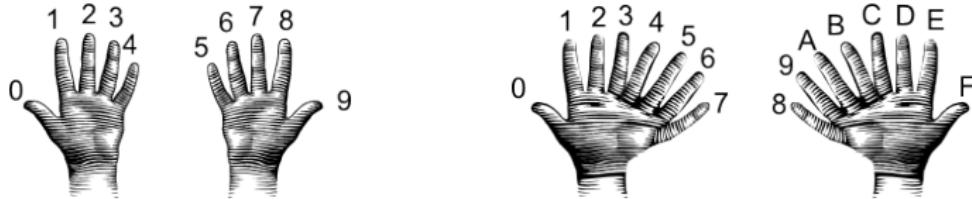
- Example: what is 99 as a decimal number?

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?

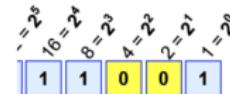
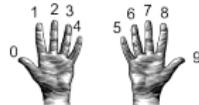
9 in decimal is 9.  $9 \times 16$  is 144.

9 in decimal digits is 9

$144 + 9$  is 153.

Answer is 153.

# Decimal to Binary: Converting Between Bases

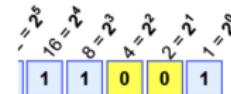
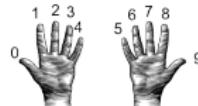


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

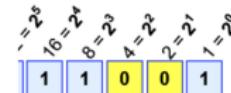
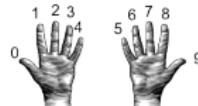


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

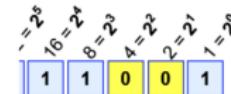
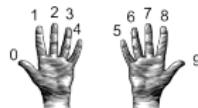


Example:  $1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

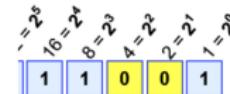
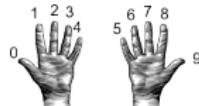


Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

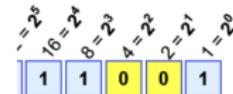


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

# Decimal to Binary: Converting Between Bases

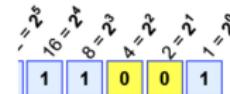
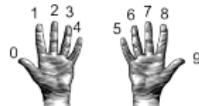


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

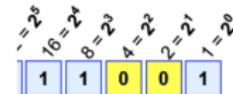
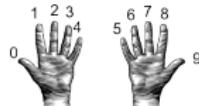


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

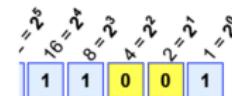
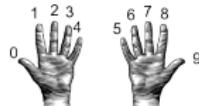


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

# Decimal to Binary: Converting Between Bases

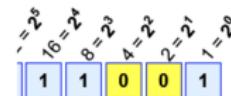
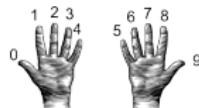


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



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

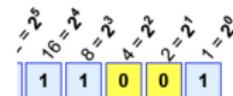
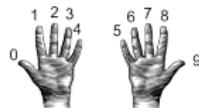
- From decimal to binary:

- 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



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

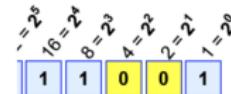
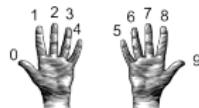
- From decimal to binary:

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

► Example: what is 130 in binary notation?

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

# Decimal to Binary: Converting Between Bases



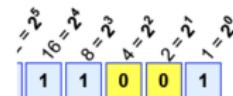
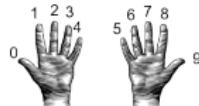
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From decimal to binary:

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

130/128 is 1 rem 2. First digit is 1: 1...  
2/64 is 0 rem 2.

# Decimal to Binary: Converting Between Bases



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

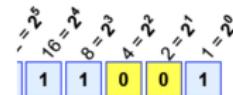
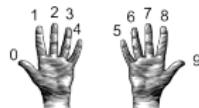
- From decimal to binary:

- 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:

# Decimal to Binary: Converting Between Bases



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

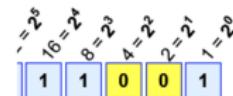
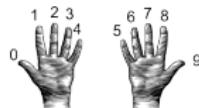
- From decimal to binary:

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

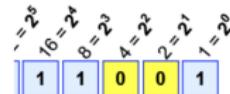
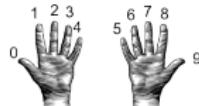
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

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

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

2/32 is 0 rem 2.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

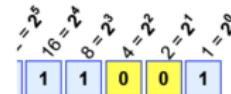
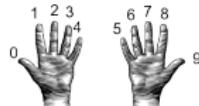
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

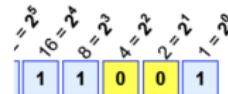
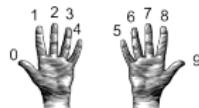
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

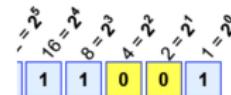
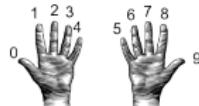
130/128 is 1 rem 2. First digit is 1: 1...

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

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

2/16 is 0 rem 2.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

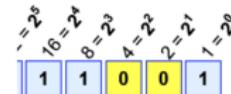
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

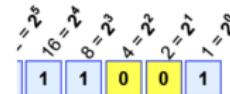
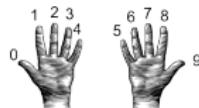
130/128 is 1 rem 2. First digit is 1: 1...

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

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

- From decimal to binary:

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

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

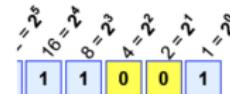
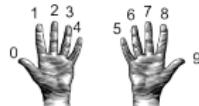
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

2/8 is 0 rem 2.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

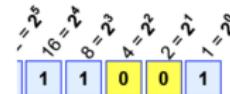
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

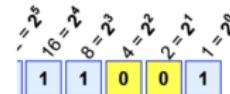
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

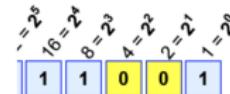
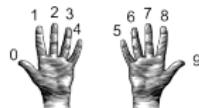
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

2/4 is 0 remainder 2.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

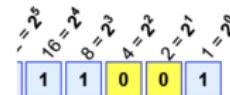
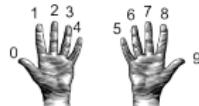
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

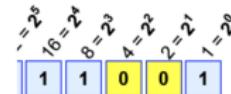
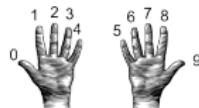
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

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

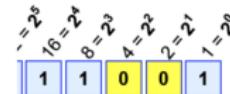
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

2/2 is 1 rem 0.

# Decimal to Binary: Converting Between Bases



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...

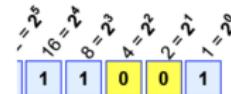
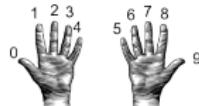
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

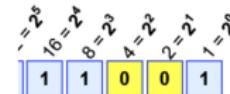
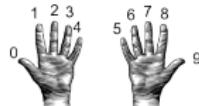
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

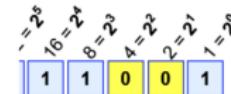
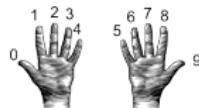
2/8 is 0 rem 2. Next digit is 0: 10000...

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



- From decimal to binary:

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

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

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

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

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

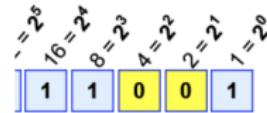
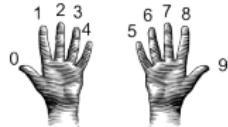
2/8 is 0 rem 2. Next digit is 0: 10000...

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

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

Adding the last remainder: 10000010

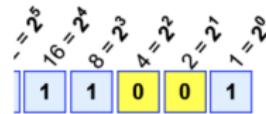
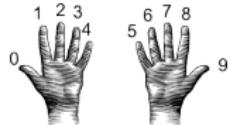
# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

# Decimal to Binary: Converting Between Bases

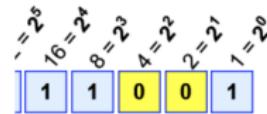


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99 / 128$  is 0 rem 99.

# Decimal to Binary: Converting Between Bases

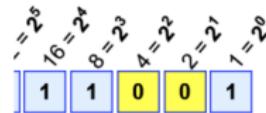


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

# Decimal to Binary: Converting Between Bases



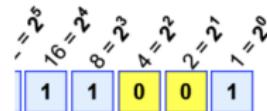
Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- Example: what is 99 in binary notation?

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

99/64 is 1 rem 35.

# Decimal to Binary: Converting Between Bases



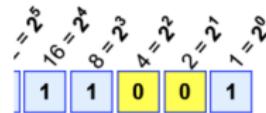
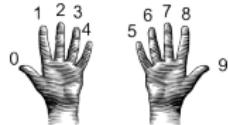
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

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

# Decimal to Binary: Converting Between Bases



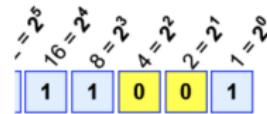
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

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

# Decimal to Binary: Converting Between Bases



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

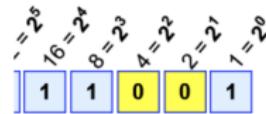
- Example: what is 99 in binary notation?

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

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

35/32 is 1 rem 3.

# Decimal to Binary: Converting Between Bases



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

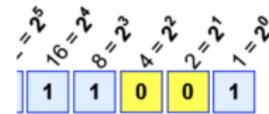
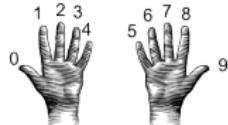
- Example: what is 99 in binary notation?

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

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

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

# Decimal to Binary: Converting Between Bases



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

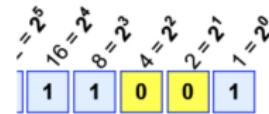
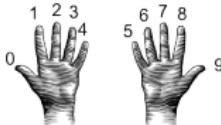
- Example: what is 99 in binary notation?

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

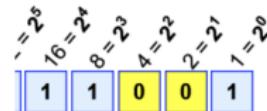
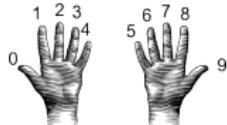
99/128 is 0 rem 99. First digit is 0: 0...

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

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

3/16 is 0 rem 3.

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

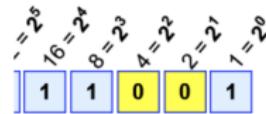
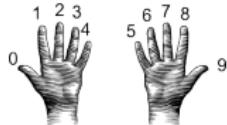
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

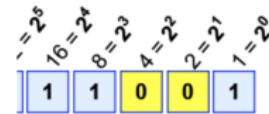
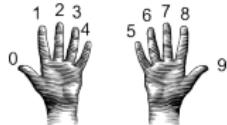
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

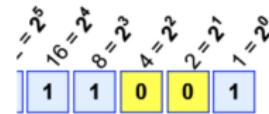
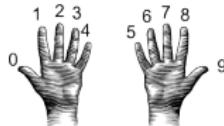
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

3/8 is 0 rem 3.

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

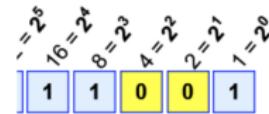
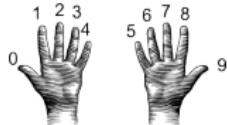
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

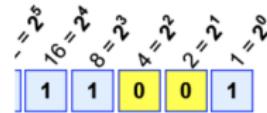
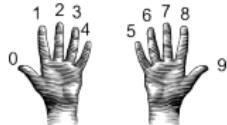
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

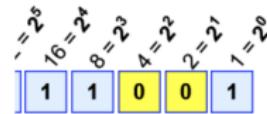
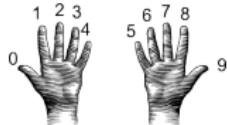
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

3/4 is 0 remainder 3.

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

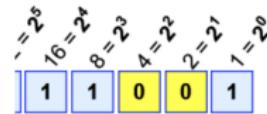
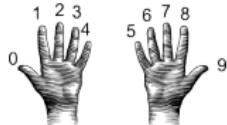
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

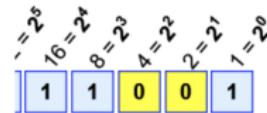
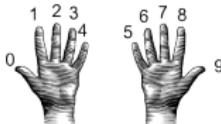
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

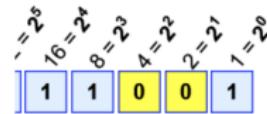
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1.

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

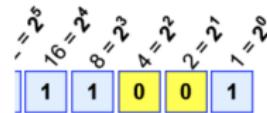
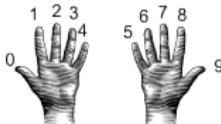
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

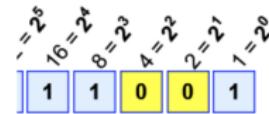
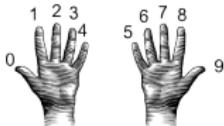
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

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

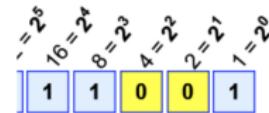
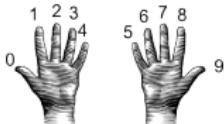
3/8 is 0 rem 3. Next digit is 0: 01100...

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

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

Adding the last remainder: 01100011

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

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

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

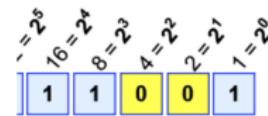
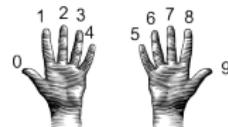
3/4 is 0 remainder 3. Next digit is 0: 011000...

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

Adding the last remainder: 01100011

Answer is 1100011.

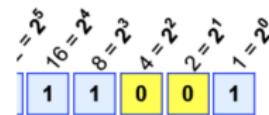
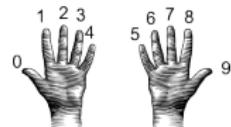
# Binary to Decimal: Converting Between Bases



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

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

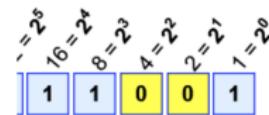
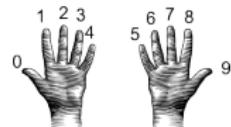
# Binary to Decimal: Converting Between Bases



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

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

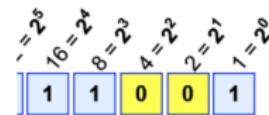
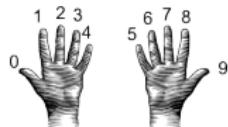
# Binary to Decimal: Converting Between Bases



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

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

# Binary to Decimal: Converting Between Bases

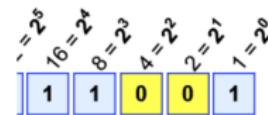
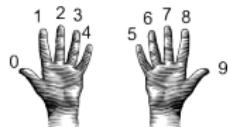


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

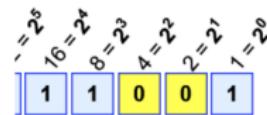
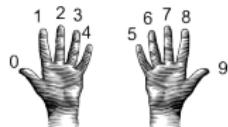


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

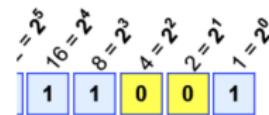
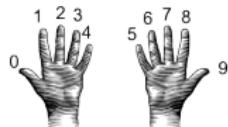


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

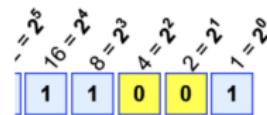
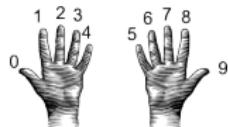


Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

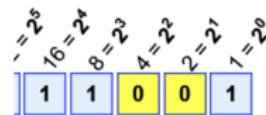
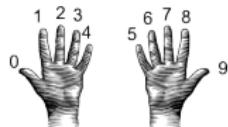


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

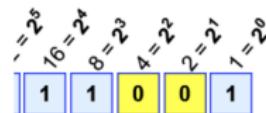
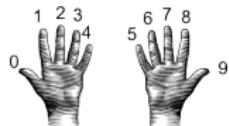


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases



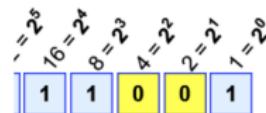
Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- From binary to decimal:

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

Sum starts with:

# Binary to Decimal: Converting Between Bases



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

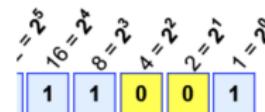
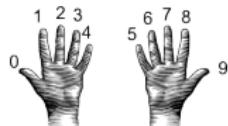
- From binary to decimal:

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

Sum starts with: 1

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

# Binary to Decimal: Converting Between Bases



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

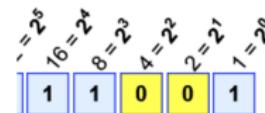
- From binary to decimal:

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

Sum starts with: 1

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

# Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

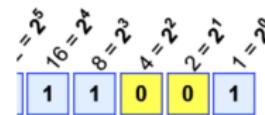
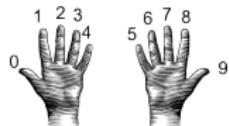
- Set sum = last digit.
- Multiply next digit by  $2^1$ . Add to sum.
- Multiply next digit by  $4 = 2^2$ . Add to sum.
- Multiply next digit by  $8 = 2^3$ . Add to sum.
- Multiply next digit by  $16 = 2^4$ . Add to sum.
- Multiply next digit by  $32 = 2^5$ . Add to sum.
- Multiply next digit by  $64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

Sum starts with: 1

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

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

# Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

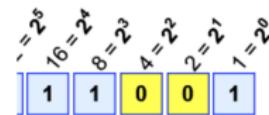
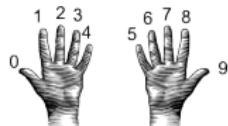
- Set sum = last digit.
- Multiply next digit by  $2 = 2^1$ . Add to sum.
- Multiply next digit by  $4 = 2^2$ . Add to sum.
- Multiply next digit by  $8 = 2^3$ . Add to sum.
- Multiply next digit by  $16 = 2^4$ . Add to sum.
- Multiply next digit by  $32 = 2^5$ . Add to sum.
- Multiply next digit by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

Sum starts with: 1

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

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

# Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

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

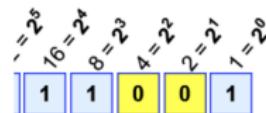
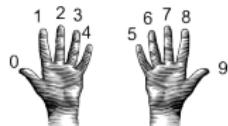
Sum starts with: 1

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

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

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

# Binary to Decimal: Converting Between Bases



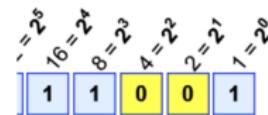
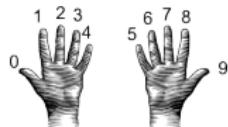
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



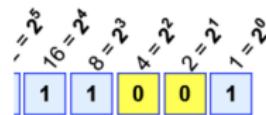
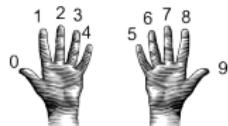
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



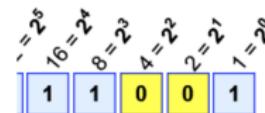
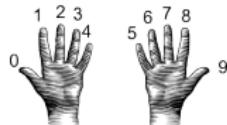
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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

# Binary to Decimal: Converting Between Bases



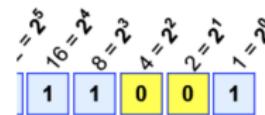
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



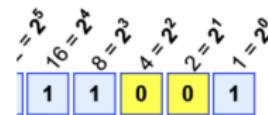
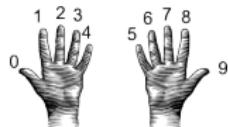
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



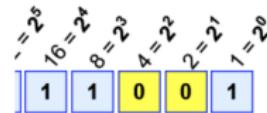
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases

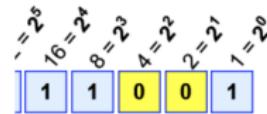
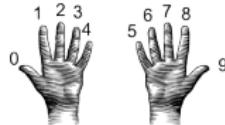


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

# Binary to Decimal: Converting Between Bases



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

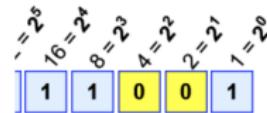
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

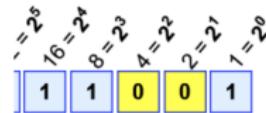
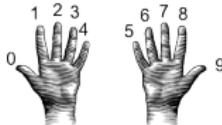
Sum starts with:

0

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

0

# Binary to Decimal: Converting Between Bases



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

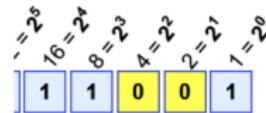
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

# Binary to Decimal: Converting Between Bases



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

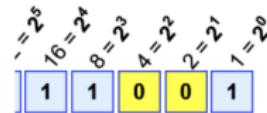
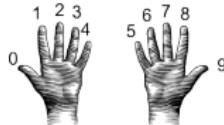
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

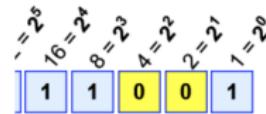
Sum starts with: 0

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

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

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

# 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?

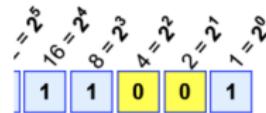
Sum starts with: 0

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

- Example: What is 10100100 in decimal?

Sum starts with: 0

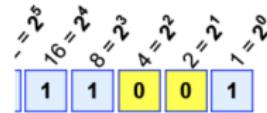
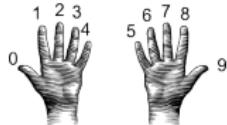
$0 \times 2 = 0$ . Add 0 to sum: 0

$1 \times 4 = 4$ . 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

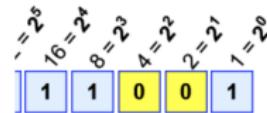
$0 \times 2 = 0$ . Add 0 to sum: 0

$1 \times 4 = 4$ . 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

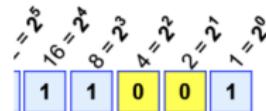
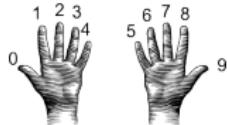
$1 \times 4 = 4$ . Add 4 to sum: 4

$0 \times 8 = 0$ . 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

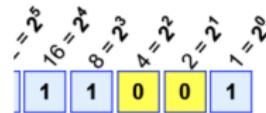


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

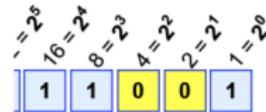
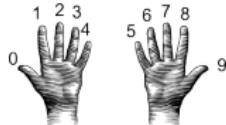
$0 \times 8 = 0$ . Add 0 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

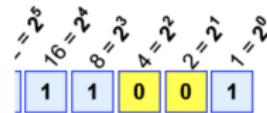
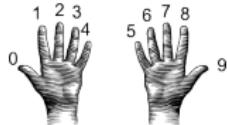
$0 \times 8 = 0$ . Add 0 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



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

- 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

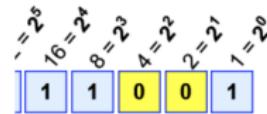
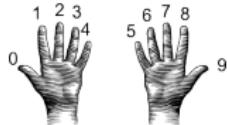
$0 \times 16 = 0$ . Add 0 to sum: 4

$1 \times 32 = 32$ . 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

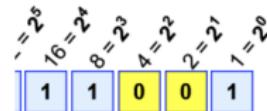
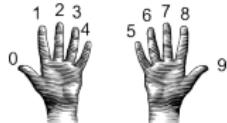


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

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

# Binary to Decimal: Converting Between Bases



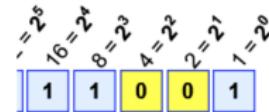
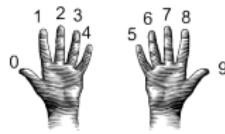
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

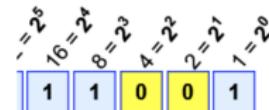
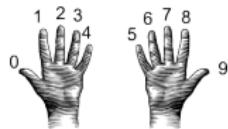
# Design Challenge: Incrementers



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

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

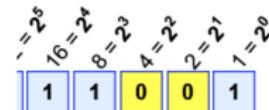
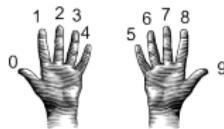
# Design Challenge: Incrementers



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

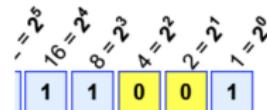
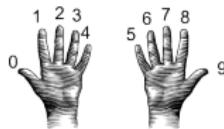


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

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

# Design Challenge: Incrementers



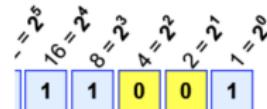
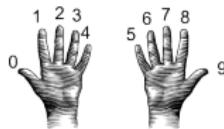
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



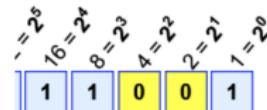
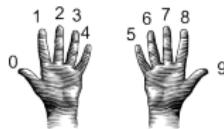
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

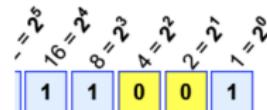
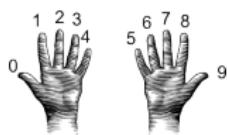
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



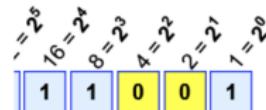
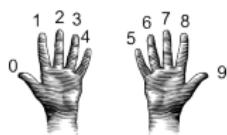
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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"  
*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.

# Today's Topics



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

# Final Overview: Administration

- The exam will be administered through Gradescope.

# Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only during the time of the exam

# Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only during the time of the exam
- There will be a different Gradescope Course called **CSci 127 Final Exam**

# Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only during the time of the exam
- There will be a different Gradescope Course called **CSci 127 Final Exam**
- Prior to the exam you will be added to the final exam course for your exam version.

# Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only during the time of the exam
- There will be a different Gradescope Course called **CSci 127 Final Exam**
- Prior to the exam you will be added to the final exam course for your exam version.
- The only assignment in that course will be your final exam.

# Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only during the time of the exam
- There will be a different Gradescope Course called **CSci 127 Final Exam**
- Prior to the exam you will be added to the final exam course for your exam version.
- The only assignment in that course will be your final exam.
- The morning of the exam: log into Gradescope, find the **CSci 127 Final Exam** course and open the assignment.

# Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.

# Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
  - ▶ With notes, examples, programs: what will help you on the exam.

# Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
  - ▶ With notes, examples, programs: what will help you on the exam.
  - ▶ Best if you design/write yours since excellent way to study.

# Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
  - ▶ With notes, examples, programs: what will help you on the exam.
  - ▶ Best if you design/write yours since excellent way to study.
  - ▶ Avoid scrambling through web searches and waste time during the exam.

# Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
  - ▶ With notes, examples, programs: what will help you on the exam.
  - ▶ Best if you design/write yours since excellent way to study.
  - ▶ Avoid scrambling through web searches and waste time during the exam.
- The exam format:

# Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
  - ▶ With notes, examples, programs: what will help you on the exam.
  - ▶ Best if you design/write yours since excellent way to study.
  - ▶ Avoid scrambling through web searches and waste time during the exam.
- The exam format:
  - ▶ Like a long Lab Quiz, you scroll down to answer all questions.

# Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
  - ▶ With notes, examples, programs: what will help you on the exam.
  - ▶ Best if you design/write yours since excellent way to study.
  - ▶ Avoid scrambling through web searches and waste time during the exam.
- The exam format:
  - ▶ Like a long Lab Quiz, you scroll down to answer all questions.
  - ▶ Questions roughly correspond to the 10 parts from old exams, but will appear as a larger number of questions on Gradescope
  - ▶ Questions are variations on the programming assignments, lab exercises, and lecture design challenges.

# Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
  - ▶ With notes, examples, programs: what will help you on the exam.
  - ▶ Best if you design/write yours since excellent way to study.
  - ▶ Avoid scrambling through web searches and waste time during the exam.
- The exam format:
  - ▶ Like a long Lab Quiz, you scroll down to answer all questions.
  - ▶ Questions roughly correspond to the 10 parts from old exams, but will appear as a larger number of questions on Gradescope
  - ▶ Questions are variations on the programming assignments, lab exercises, and lecture design challenges.
- Past exams available on webpage (includes answer keys).

# Exam Options

## Exam Times:

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

19 December 2008

### 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 contains notes, programs, or formulas.
- You may take the exam, you may have up to one pen and pencil, and your note sheet.
- You are not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, obtaining other advantage, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The student handbook of the CUNY system provides more information about the consequences of academic honesty and will pursue cases of academic dishonesty according to the Hunter College Academic Honesty Policy.

I acknowledge that all cases of academic dishonesty will be reported to the Office of Student and Academic Conduct.	
Name:	
Social Security:	
Email:	
Signature:	

# Exam Options

## Exam Times:

- Default: Regular Time: Monday, 14 December, 9-11am.

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.
- All work, notes, programs, and other materials must be handwritten.
- You may bring a calculator, you may have up to one pen and pencil, and your note sheet.
- You are not to use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, obtaining other advantage, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The student handbook at CUNY.org provides more information about what constitutes academic dishonesty according to the Hunter College Student Handbook.

I declare that all work on this exam is my own work and not plagiarized in any way.	
Name:	
Signature:	
Email:	
Signature:	

# Exam Options

## Exam Times:

- Default: Regular Time: Monday, 14 December, 9-11am.
- Alternate Time: Reading Day, Friday, 11 December, 8am-10am.

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

19 December 2008

### 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 1/2" x 11" piece of paper that contains pre-arranged formulas.
- You may take breaks; you may have water, pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, obtaining other advantage, and fabrication of records and official documents) as serious offenses that violate the principles of the University. The University's policy on CREDIT FOR ACTS OF ACADEMIC DISHONESTY AND PUNISHMENT OF STUDENTS FOR ACTS OF ACADEMIC DISHONESTY

I acknowledge that all forms of academic dishonesty will be reported to the Office of Student and Academic Conduct.	
Name:	
Signature:	
Email:	
Signature:	

# Exam Options

## Exam Times:

- Default: Regular Time: Monday, 14 December, 9-11am.
- Alternate Time: Reading Day, Friday, 11 December, 8am-10am.
- Accessibility Testing: If you have not done so already, email me no later than 7 December.

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

19 December 2020

### 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.
- Use a scientific calculator if you need one.
- You may bring a calculator, pen and pencil, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, obtaining other advantage, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The student handbook of the CUNY system provides more information about the consequences of academic honesty and will pursue cases of academic dishonesty according to the Hunter College Student Handbook.

I acknowledge that all work on this exam is my own and is equivalent to the work I submitted and will submit in my class.	
Name:	
Signature:	
Email:	
Signature:	

# Exam Options

## Exam Times:

- Default: Regular Time: Monday, 14 December, 9-11am.
- Alternate Time: Reading Day, Friday, 11 December, 8am-10am.
- Accessibility Testing: If you have not done so already, email me no later than 7 December.
- Survey for your choice will be available next lecture. **No survey answer implies you will take the exam on 14 December.**

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

19 December 2020

### 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 contains programs, formulas, or other relevant information. This is to prevent CSCI 127 from being accused of academic integrity violations.
- You may bring into the exam room your pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, obtaining other advantage, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The Hunter College Code of Conduct governs academic honesty and will pursue cases of academic dishonesty according to the Hunter College Code of Conduct.

I understand that acts of academic dishonesty will be reported to the Office of Student and Academic Conduct.	
Name:	
Social Security:	
Email:	
Signature:	

# Exam Options

## Exam Times:

- Default: Regular Time: Monday, 14 December, 9-11am.
- Alternate Time: Reading Day, Friday, 11 December, 8am-10am.
- Accessibility Testing: If you have not done so already, email me no later than 7 December.
- Survey for your choice will be available next lecture. **No survey answer implies you will take the exam on 14 December.**

## Grading Options:

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

19 December 2020

### 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 contains prewritten formulas and constants. This is the same paper used for CSci 127.
- You may bring a calculator, ruler, compass, protractor, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, obtaining other advantage, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The Hunter College Code of Conduct defines academic honesty and will pursue cases of academic dishonesty according to the Hunter College Code of Conduct.

I understand that acts of academic dishonesty will be reported to the Office of Student and Academic Conduct.	
Name:	
Signature:	
Email:	
Signature:	

# Exam Options

## Exam Times:

- Default: Regular Time: Monday, 14 December, 9-11am.
- Alternate Time: Reading Day, Friday, 11 December, 8am-10am.
- Accessibility Testing: If you have not done so already, email me no later than 7 December.
- Survey for your choice will be available next lecture. **No survey answer implies you will take the exam on 14 December.**

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

19 December 2020

### 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 contains programs and formulas. This is a CUNY-wide rule. No calculators, phones, or other electronic devices.
- You may bring in, you may have in, or you pass, 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 (i.e., plagiarism, cheating or communism, obtaining other advantage, and fabrication of records and official documents) as serious offenses against the integrity of the educational process. The student handbook provides details on academic honesty and will pursue cases of academic dishonesty according to the Hunter College Student Handbook.

I understand that all forms of academic dishonesty will be reported to the Office of Student and Academic Conduct.	
Name:	
Social Security:	
Email:	
Signature:	

## Grading Options:

- Default: Letter Grade.
- Credit/NoCredit grade— check with academic advisor and fill out form by **25 November**

# Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

# Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend [live Lab Review on Wednesday 11-12:30pm](#)

# Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend [live Lab Review on Wednesday 11-12:30pm](#)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday

# Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend [live Lab Review on Wednesday 11-12:30pm](#)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 3 programming assignments (programs 50-52)

# Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend [live Lab Review on Wednesday 11-12:30pm](#)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 3 programming assignments (programs 50-52)
- At any point, visit our [Drop-In Tutoring 11am-5pm](#) for help!!!

# Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend [live Lab Review on Wednesday 11-12:30pm](#)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 3 programming assignments (programs 50-52)
- At any point, visit our [Drop-In Tutoring 11am-5pm](#) for help!!!
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)