

# CSci 127: Introduction to Computer Science



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

# 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 Lectures in 10 Minutes



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

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

# Lecture 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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

- As well as definite loops & the turtle package:

The screenshot shows a Python code editor with a file named 'main.py' open. The code is as follows:

```
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 'Result' window displaying a purple hexagon drawn by the turtle. The turtle has stamped a purple star at each vertex of the hexagon.

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

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

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

## Lecture 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

## Lecture 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

## Lecture 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

## Lecture 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

## Lecture 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']

## Lecture 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.

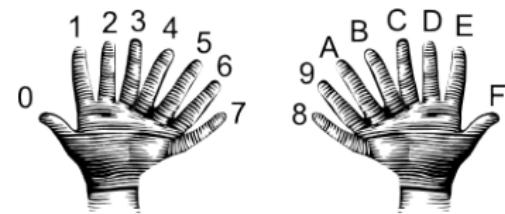
## Lecture 2: variables, data types, more on loops & range()

- 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)
```

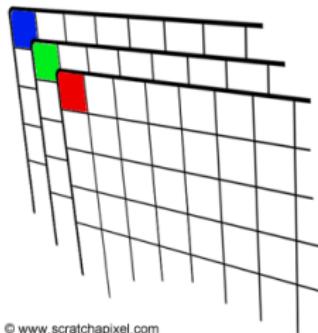
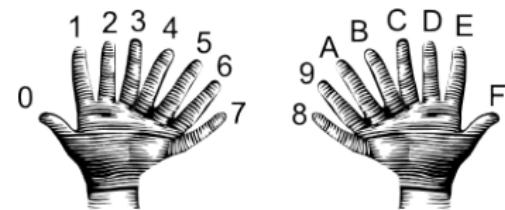
# Lecture 3: colors, hex, slices, numpy & images

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



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

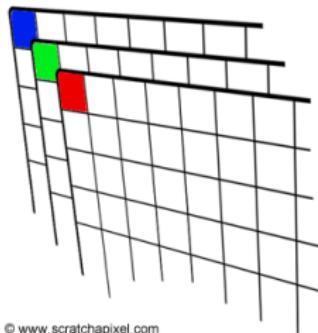
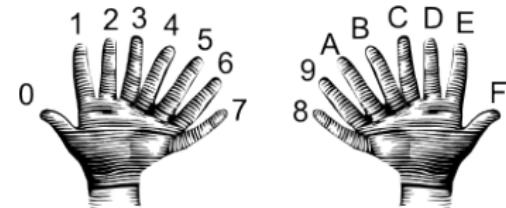
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



© www.scratchapixel.com

# Lecture 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

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



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



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

# Lecture 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.

# Lecture 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.

## Lecture 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')
```

# Lecture 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!")
```

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

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

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

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



# Lecture 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,Totals  
1890,4937,2037,,727,7181  
1871,21843,3623,,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,63545,5540,6442,1755,4543,75934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,11013,14035,5346,10965,391114  
1850,35544,12891,18951,5346,10965,391115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911801  
1890,1367711,66582,11582,51980,33029,2141514  
1900,185093,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,4766803  
1920,22331103,2018354,44607,73201,11651,550488  
1930,26671103,2018354,44607,73201,11651,550446  
1940,1889924,24690285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690101,2738175,1899049,1451277,191555,7981984  
1970,1539231,24690285,1471701,139443,7981984  
1980,1428285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322646  
2000,1537195,2485326,2229379,1332650,419782,8080879  
2010,1583873,2504705,2277722,1385108,447558,8155133  
2015,1444518,2646733,2339150,1454446,474558,8056405

nycHistPop.csv

In Lab 6

# Lecture 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  
1771,21843,36231,,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67541,6240,6842,1755,4543,75934  
1820,123704,11487,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14034,5346,10965,391114  
1850,355441,21803,18891,5346,10965,391115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911801  
1890,1387511,71041,6348,51980,33029,1911804  
1900,185093,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,4766803  
1920,22101103,2018354,44671,44671,73201,11651,50048  
1930,16671112,1579128,1579128,1579128,1579128,4930446  
1940,1889924,2469285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690111,1809239,1809239,1809239,1809239,781984  
1970,1539231,1867011,1867011,1867011,1867011,781984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2223379,1332450,419728,8080879  
2010,1583873,2504705,2272722,1385108,419728,8175133  
2015,1444518,2646733,2339150,1459446,474558,8056405
```

nycHistPop.csv

In Lab 6

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

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,2037,727,788,102  
1771,21843,3623,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67530,6240,6840,2000,4973,85934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3032,7082,242278  
1840,311510,19013,14045,5346,10965,391114  
1850,355441,21800,18895,6254,11515,50115  
1860,813469,279122,32903,23593,25492,217477  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1370000,710000,65000,58000,35000,2150000  
1900,1850593,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,2210110,2018354,44601,44601,73201,11651,50048  
1930,1867112,1797128,1797128,1797128,15821,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690211,2319319,1890911,1460711,202011,781984  
1970,1539231,2465701,1471071,135443,796462  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,73977,7322564  
2000,1537195,2485326,2229379,1332650,419782,8080879  
2010,1583873,2504705,2277722,1385108,8175133  
2015,1444518,2436733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6

# Lecture 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  
1690,4937,2017,...,727,7181  
1771,21843,36241,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,61000,68000,18000,49734  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,311510,11013,14000,5348,10965,391114  
1850,355400,128000,185000,5348,10965,391114  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59940,5653,51980,33029,1911801  
1890,1380000,710000,680000,51980,33029,1911801  
1900,1850993,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,4766803  
1920,22161103,2018354,44600,720201,11651,50048  
1930,19671128,1797128,200000,720201,11651,50048  
1940,1889924,2469285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1696000,2319319,1809000,1451277,191555,7981984  
1970,1539231,2460701,1472701,1235443,7981984  
1980,1428285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,379977,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,474558,8175133  
2015,1444518,2646733,2339150,1459446,474558,8056405
```

nycHistPop.csv

In Lab 6

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

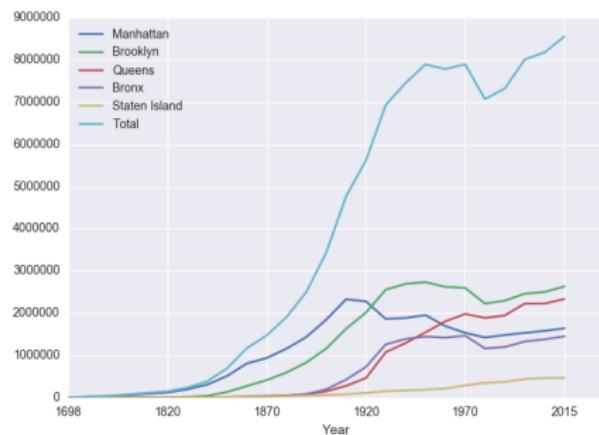
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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  
1810,71531,6200,6800,1800,4973,85934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,19013,14000,5348,10965,391114  
1850,435441,21800,18500,5850,13000,50115  
1860,613469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1385711,72000,63000,58000,35000,210134  
1900,1850993,116582,152999,200567,67921,2437202  
1910,233142,1634351,2841,430980,8569,476683  
1920,2210103,2018354,44601,73201,11600,50048  
1930,2667128,2407128,47000,72054,15820,5930446  
1940,188924,2469285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7991957  
1960,1690211,2303219,180900,140000,120000,781984  
1970,1539231,2465071,1472701,1235443,798462  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2216722,1385108,450750,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6



# Lecture 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()
```

# Lecture 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.

# Lecture 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()`

# Lecture 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:

# Lecture 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")`

# Lecture 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,

# Lecture 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.

# Lecture 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)
```

# Lecture 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).

# Lecture 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**.

# Lecture 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**

# Lecture 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.

# Lecture 8: function parameters, github

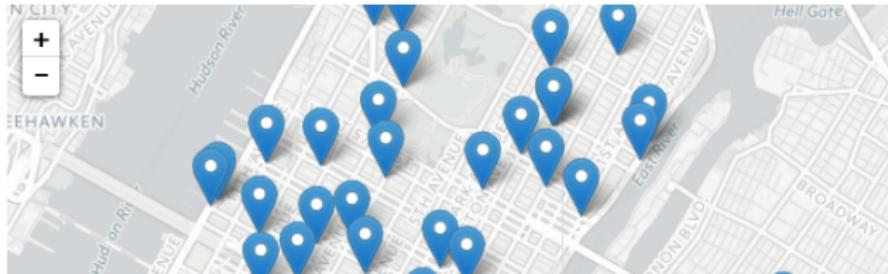
```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

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

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

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

# Lecture 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)
```

# Lecture 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)
```

# Lecture 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)
```

# Lecture 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.

# Lecture 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.`

# Lecture 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 Lectures in 10 Minutes



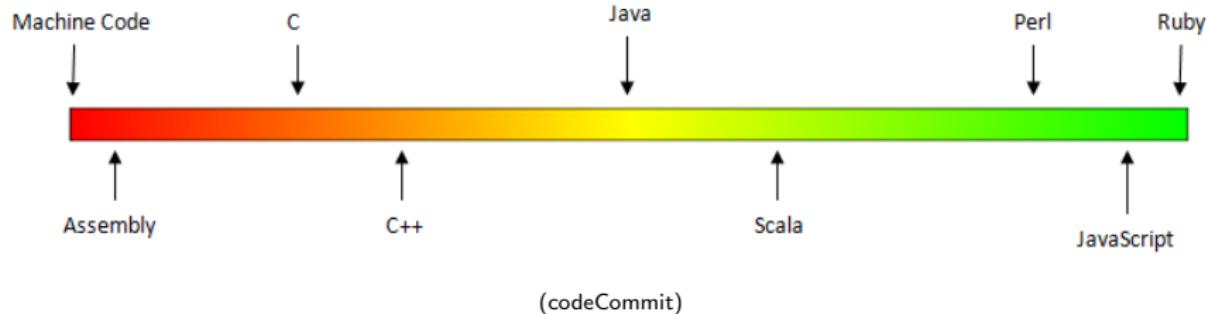
- Input/Output (I/O): `input()` and `print()`; pandas for CSV files
- Types:
  - ▶ Primitive: `int`, `float`, `bool`, `string`;
  - ▶ Container: lists (but not dictionaries/hashes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`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

# Today's Topics



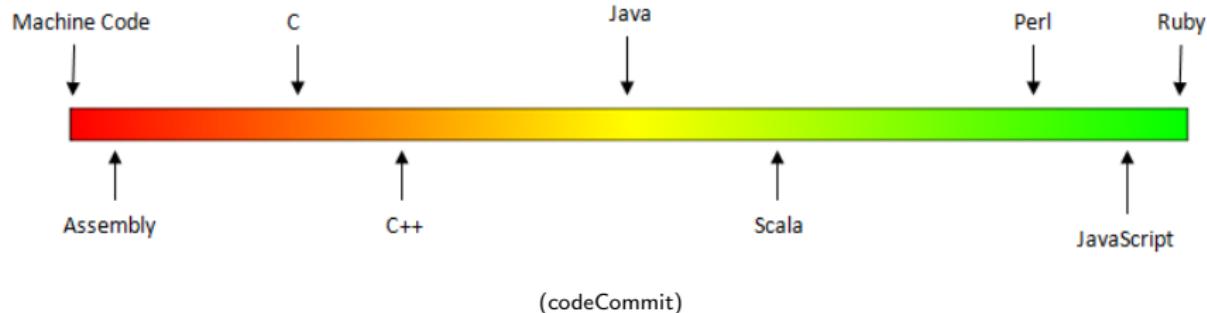
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

# Low-Level vs. High-Level Languages



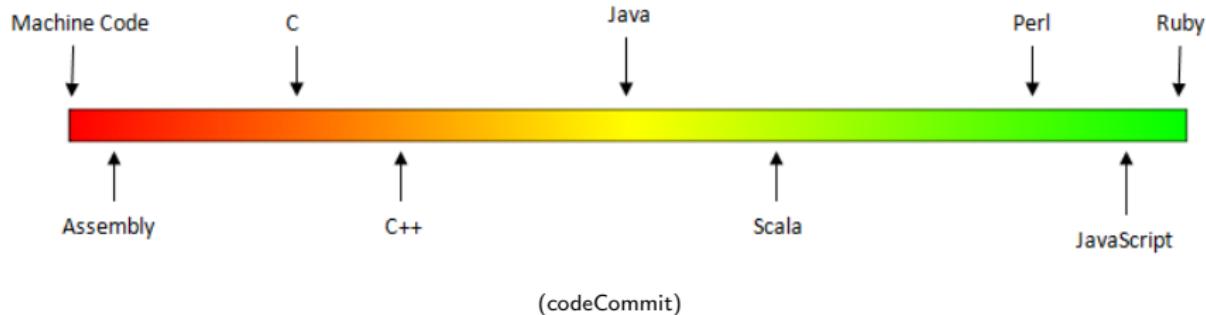
- Can view programming languages on a continuum.

# Low-Level vs. High-Level Languages



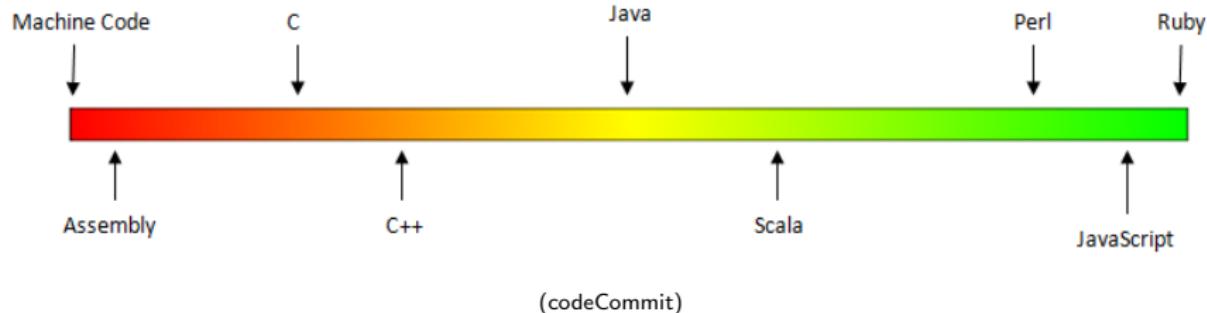
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

# Low-Level vs. High-Level Languages



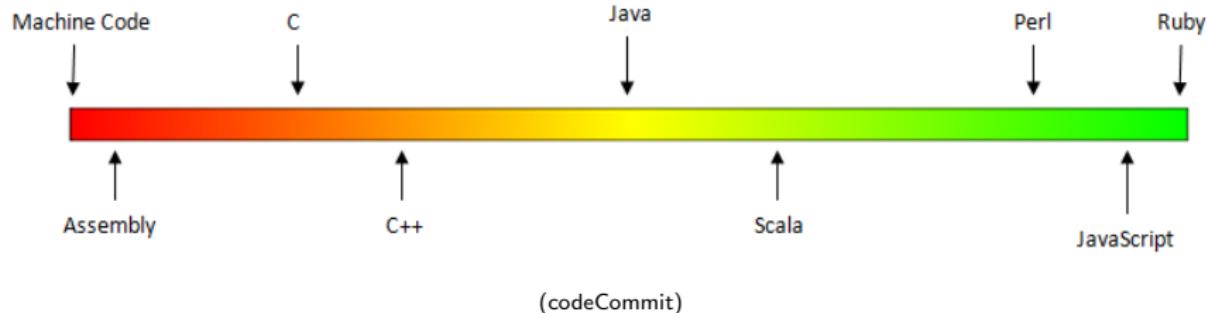
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

# Low-Level vs. High-Level Languages



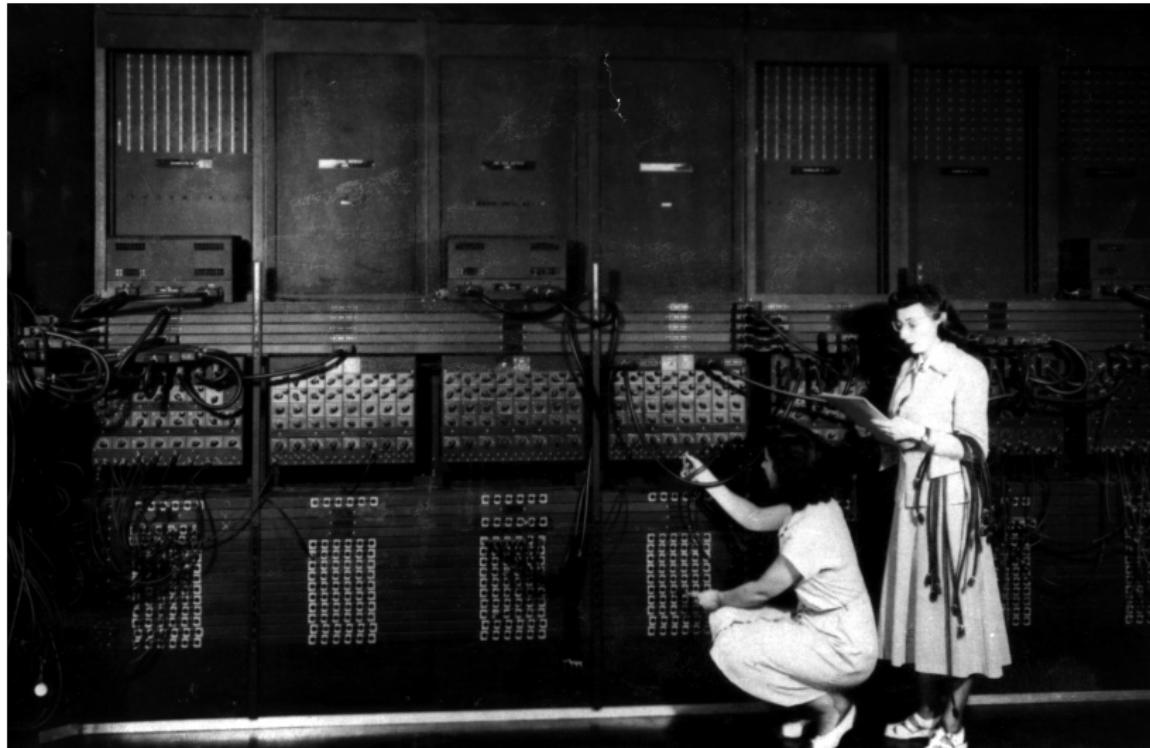
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine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

# Machine Language



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

# Machine Language

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

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

BREAK

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

(wiki)

# Machine Language

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

```

A 002000 C2 3B REP #3B
A 002002 SED
A 002003 CLC
A 002004 FS
A 002005 LD #1234
A 002006 LD #12345678
A 002007 LD #12345678
A 002008 LD #12345678
A 002009 LD #12345678
A 00200A LD #12345678
A 00200B LD #12345678
A 00200C LD #12345678
A 00200D LD #12345678
A 00200E LD #12345678
A 00200F LD #12345678
A 002011 RR
A 002012 RR

F PB PC Mm#012C .A X Y SP DP IB
; 00 2012 00110000 0000 0000 0002 CF77 0000 00
; 2000

BREAK

PB PC Mm#012C .A X Y SP DP IB
; 00 2013 00110000 0555 0000 0002 CF77 0000 00
; 7703 7703

n 7703 7703

```

(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.

```

A 002000 C2 36 REP #430
A 002022 FB CLC
A 002043 FB SED
A 002063 F0 14 LSH #01234
A 002087 73 45 ADD #0145
A 0020A8 FF 81 STA #017789
A 0020C6 D0 CLD
A 0020E7 E2 36 SWP #38
A 002111 98 BHK
A 002122
F PB PC Min#012C A X Y SP DP IB
; 00 2013 00110800 0550 0000 0002 CF77 0000 00
$ 2000
BREAK

PB PC Min#012C A X Y SP DP IB
; 00 2013 00110800 0550 0000 0002 CF77 0000 00
$ 7793 7793
n 7793 7793
m 7793 7793
l 7793 7793

```

(wiki)

# Machine Language

(wiki)

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

# Machine Language

(wiki)

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

# “Hello World!” in Simplified Machine Language

Line: 3 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

Step	Run	<input checked="" type="checkbox"/> Enable auto switching			
S	T	A	V	Stack	Log
s0:				10	
s1:				9	
s2:				9	
s3:				22	
s4:				696	
s5:				976	
s6:				927	
s7:				418	

(WeMIPS)

# WeMIPS

The screenshot shows the WeMIPS IDE interface. At the top, there are tabs for 'Run', 'Step', and 'Run' (with a checked checkbox for 'Enable auto switching'). Below the tabs are buttons for 'Addition Doubler', 'Stax', 'Looper', 'Stack Test', and 'Hello World'. Further down are buttons for 'Code Gen Save String', 'Interactive', 'Binary2 Decimal', and 'Decimal2 Binary'. A 'Debug' button is also present.

The main area displays assembly code:

```
# Store 'Hello world!' at the top of the stack
1    .text
2    .globl _start
3    .section .text
4    _start:
5        .ascii "Hello world!\n"
6        .size _start, . - _start
7        .section .data
8        .space 100
9        .section .bss
10       .space 100
11
12        .text
13        .globl _main
14        .section .text
15        _main:
16            .size _main, . - _main
17
18        .text
19        .globl _exit
20        .section .text
21        _exit:
22            .size _exit, . - _exit
23
24        .text
25        .globl _printf
26        .section .text
27        _printf:
28            .size _printf, . - _printf
29
30        .text
31        .globl _main
32        .section .text
33        _main:
34            .size _main, . - _main
35
36        .text
37        .globl _main
38        .section .text
39        _main:
40            .size _main, . - _main
41
42        .text
43        .globl _main
44        .section .text
45        _main:
46            .size _main, . - _main
47
48        .text
49        .globl _main
50        .section .text
51        _main:
52            .size _main, . - _main
53
54        .text
55        .globl _main
56        .section .text
57        _main:
58            .size _main, . - _main
59
60        .text
61        .globl _main
62        .section .text
63        _main:
64            .size _main, . - _main
65
66        .text
67        .globl _main
68        .section .text
69        _main:
70            .size _main, . - _main
71
72        .text
73        .globl _main
74        .section .text
75        _main:
76            .size _main, . - _main
77
78        .text
79        .globl _main
80        .section .text
81        _main:
82            .size _main, . - _main
83
84        .text
85        .globl _main
86        .section .text
87        _main:
88            .size _main, . - _main
89
90        .text
91        .globl _main
92        .section .text
93        _main:
94            .size _main, . - _main
95
96        .text
97        .globl _main
98        .section .text
99        _main:
100       .size _main, . - _main
101
102       .text
103       .globl _main
104       .section .text
105       _main:
106           .size _main, . - _main
107
108       .text
109       .globl _main
110       .section .text
111       _main:
112           .size _main, . - _main
113
114       .text
115       .globl _main
116       .section .text
117       _main:
118           .size _main, . - _main
119
120       .text
121       .globl _main
122       .section .text
123       _main:
124           .size _main, . - _main
125
126       .text
127       .globl _main
128       .section .text
129       _main:
130           .size _main, . - _main
131
132       .text
133       .globl _main
134       .section .text
135       _main:
136           .size _main, . - _main
137
138       .text
139       .globl _main
140       .section .text
141       _main:
142           .size _main, . - _main
143
144       .text
145       .globl _main
146       .section .text
147       _main:
148           .size _main, . - _main
149
150       .text
151       .globl _main
152       .section .text
153       _main:
154           .size _main, . - _main
155
156       .text
157       .globl _main
158       .section .text
159       _main:
160           .size _main, . - _main
161
162       .text
163       .globl _main
164       .section .text
165       _main:
166           .size _main, . - _main
167
168       .text
169       .globl _main
170       .section .text
171       _main:
172           .size _main, . - _main
173
174       .text
175       .globl _main
176       .section .text
177       _main:
178           .size _main, . - _main
179
180       .text
181       .globl _main
182       .section .text
183       _main:
184           .size _main, . - _main
185
186       .text
187       .globl _main
188       .section .text
189       _main:
190           .size _main, . - _main
191
192       .text
193       .globl _main
194       .section .text
195       _main:
196           .size _main, . - _main
197
198       .text
199       .globl _main
200       .section .text
201       _main:
202           .size _main, . - _main
203
204       .text
205       .globl _main
206       .section .text
207       _main:
208           .size _main, . - _main
209
210       .text
211       .globl _main
212       .section .text
213       _main:
214           .size _main, . - _main
215
216       .text
217       .globl _main
218       .section .text
219       _main:
220           .size _main, . - _main
221
222       .text
223       .globl _main
224       .section .text
225       _main:
226           .size _main, . - _main
227
228       .text
229       .globl _main
230       .section .text
231       _main:
232           .size _main, . - _main
233
234       .text
235       .globl _main
236       .section .text
237       _main:
238           .size _main, . - _main
239
240       .text
241       .globl _main
242       .section .text
243       _main:
244           .size _main, . - _main
245
246       .text
247       .globl _main
248       .section .text
249       _main:
250           .size _main, . - _main
251
252       .text
253       .globl _main
254       .section .text
255       _main:
256           .size _main, . - _main
257
258       .text
259       .globl _main
260       .section .text
261       _main:
262           .size _main, . - _main
263
264       .text
265       .globl _main
266       .section .text
267       _main:
268           .size _main, . - _main
269
270       .text
271       .globl _main
272       .section .text
273       _main:
274           .size _main, . - _main
275
276       .text
277       .globl _main
278       .section .text
279       _main:
280           .size _main, . - _main
281
282       .text
283       .globl _main
284       .section .text
285       _main:
286           .size _main, . - _main
287
288       .text
289       .globl _main
290       .section .text
291       _main:
292           .size _main, . - _main
293
294       .text
295       .globl _main
296       .section .text
297       _main:
298           .size _main, . - _main
299
299       .text
300       .globl _main
301       .section .text
302       _main:
303           .size _main, . - _main
304
304       .text
305       .globl _main
306       .section .text
307       _main:
308           .size _main, . - _main
309
309       .text
310       .globl _main
311       .section .text
312       _main:
313           .size _main, . - _main
314
314       .text
315       .globl _main
316       .section .text
317       _main:
318           .size _main, . - _main
319
319       .text
320       .globl _main
321       .section .text
322       _main:
323           .size _main, . - _main
324
324       .text
325       .globl _main
326       .section .text
327       _main:
328           .size _main, . - _main
329
329       .text
330       .globl _main
331       .section .text
332       _main:
333           .size _main, . - _main
334
334       .text
335       .globl _main
336       .section .text
337       _main:
338           .size _main, . - _main
339
339       .text
340       .globl _main
341       .section .text
342       _main:
343           .size _main, . - _main
344
344       .text
345       .globl _main
346       .section .text
347       _main:
348           .size _main, . - _main
349
349       .text
350       .globl _main
351       .section .text
352       _main:
353           .size _main, . - _main
354
354       .text
355       .globl _main
356       .section .text
357       _main:
358           .size _main, . - _main
359
359       .text
360       .globl _main
361       .section .text
362       _main:
363           .size _main, . - _main
364
364       .text
365       .globl _main
366       .section .text
367       _main:
368           .size _main, . - _main
369
369       .text
370       .globl _main
371       .section .text
372       _main:
373           .size _main, . - _main
374
374       .text
375       .globl _main
376       .section .text
377       _main:
378           .size _main, . - _main
379
379       .text
380       .globl _main
381       .section .text
382       _main:
383           .size _main, . - _main
384
384       .text
385       .globl _main
386       .section .text
387       _main:
388           .size _main, . - _main
389
389       .text
390       .globl _main
391       .section .text
392       _main:
393           .size _main, . - _main
394
394       .text
395       .globl _main
396       .section .text
397       _main:
398           .size _main, . - _main
399
399       .text
400       .globl _main
401       .section .text
402       _main:
403           .size _main, . - _main
404
404       .text
405       .globl _main
406       .section .text
407       _main:
408           .size _main, . - _main
409
409       .text
410       .globl _main
411       .section .text
412       _main:
413           .size _main, . - _main
414
414       .text
415       .globl _main
416       .section .text
417       _main:
418           .size _main, . - _main
419
419       .text
420       .globl _main
421       .section .text
422       _main:
423           .size _main, . - _main
424
424       .text
425       .globl _main
426       .section .text
427       _main:
428           .size _main, . - _main
429
429       .text
430       .globl _main
431       .section .text
432       _main:
433           .size _main, . - _main
434
434       .text
435       .globl _main
436       .section .text
437       _main:
438           .size _main, . - _main
439
439       .text
440       .globl _main
441       .section .text
442       _main:
443           .size _main, . - _main
444
444       .text
445       .globl _main
446       .section .text
447       _main:
448           .size _main, . - _main
449
449       .text
450       .globl _main
451       .section .text
452       _main:
453           .size _main, . - _main
454
454       .text
455       .globl _main
456       .section .text
457       _main:
458           .size _main, . - _main
459
459       .text
460       .globl _main
461       .section .text
462       _main:
463           .size _main, . - _main
464
464       .text
465       .globl _main
466       .section .text
467       _main:
468           .size _main, . - _main
469
469       .text
470       .globl _main
471       .section .text
472       _main:
473           .size _main, . - _main
474
474       .text
475       .globl _main
476       .section .text
477       _main:
478           .size _main, . - _main
479
479       .text
480       .globl _main
481       .section .text
482       _main:
483           .size _main, . - _main
484
484       .text
485       .globl _main
486       .section .text
487       _main:
488           .size _main, . - _main
489
489       .text
490       .globl _main
491       .section .text
492       _main:
493           .size _main, . - _main
494
494       .text
495       .globl _main
496       .section .text
497       _main:
498           .size _main, . - _main
499
499       .text
500       .globl _main
501       .section .text
502       _main:
503           .size _main, . - _main
504
504       .text
505       .globl _main
506       .section .text
507       _main:
508           .size _main, . - _main
509
509       .text
510       .globl _main
511       .section .text
512       _main:
513           .size _main, . - _main
514
514       .text
515       .globl _main
516       .section .text
517       _main:
518           .size _main, . - _main
519
519       .text
520       .globl _main
521       .section .text
522       _main:
523           .size _main, . - _main
524
524       .text
525       .globl _main
526       .section .text
527       _main:
528           .size _main, . - _main
529
529       .text
530       .globl _main
531       .section .text
532       _main:
533           .size _main, . - _main
534
534       .text
535       .globl _main
536       .section .text
537       _main:
538           .size _main, . - _main
539
539       .text
540       .globl _main
541       .section .text
542       _main:
543           .size _main, . - _main
544
544       .text
545       .globl _main
546       .section .text
547       _main:
548           .size _main, . - _main
549
549       .text
550       .globl _main
551       .section .text
552       _main:
553           .size _main, . - _main
554
554       .text
555       .globl _main
556       .section .text
557       _main:
558           .size _main, . - _main
559
559       .text
560       .globl _main
561       .section .text
562       _main:
563           .size _main, . - _main
564
564       .text
565       .globl _main
566       .section .text
567       _main:
568           .size _main, . - _main
569
569       .text
570       .globl _main
571       .section .text
572       _main:
573           .size _main, . - _main
574
574       .text
575       .globl _main
576       .section .text
577       _main:
578           .size _main, . - _main
579
579       .text
580       .globl _main
581       .section .text
582       _main:
583           .size _main, . - _main
584
584       .text
585       .globl _main
586       .section .text
587       _main:
588           .size _main, . - _main
589
589       .text
590       .globl _main
591       .section .text
592       _main:
593           .size _main, . - _main
594
594       .text
595       .globl _main
596       .section .text
597       _main:
598           .size _main, . - _main
599
599       .text
600       .globl _main
601       .section .text
602       _main:
603           .size _main, . - _main
604
604       .text
605       .globl _main
606       .section .text
607       _main:
608           .size _main, . - _main
609
609       .text
610       .globl _main
611       .section .text
612       _main:
613           .size _main, . - _main
614
614       .text
615       .globl _main
616       .section .text
617       _main:
618           .size _main, . - _main
619
619       .text
620       .globl _main
621       .section .text
622       _main:
623           .size _main, . - _main
624
624       .text
625       .globl _main
626       .section .text
627       _main:
628           .size _main, . - _main
629
629       .text
630       .globl _main
631       .section .text
632       _main:
633           .size _main, . - _main
634
634       .text
635       .globl _main
636       .section .text
637       _main:
638           .size _main, . - _main
639
639       .text
640       .globl _main
641       .section .text
642       _main:
643           .size _main, . - _main
644
644       .text
645       .globl _main
646       .section .text
647       _main:
648           .size _main, . - _main
649
649       .text
650       .globl _main
651       .section .text
652       _main:
653           .size _main, . - _main
654
654       .text
655       .globl _main
656       .section .text
657       _main:
658           .size _main, . - _main
659
659       .text
660       .globl _main
661       .section .text
662       _main:
663           .size _main, . - _main
664
664       .text
665       .globl _main
666       .section .text
667       _main:
668           .size _main, . - _main
669
669       .text
670       .globl _main
671       .section .text
672       _main:
673           .size _main, . - _main
674
674       .text
675       .globl _main
676       .section .text
677       _main:
678           .size _main, . - _main
679
679       .text
680       .globl _main
681       .section .text
682       _main:
683           .size _main, . - _main
684
684       .text
685       .globl _main
686       .section .text
687       _main:
688           .size _main, . - _main
689
689       .text
690       .globl _main
691       .section .text
692       _main:
693           .size _main, . - _main
694
694       .text
695       .globl _main
696       .section .text
697       _main:
698           .size _main, . - _main
699
699       .text
700       .globl _main
701       .section .text
702       _main:
703           .size _main, . - _main
704
704       .text
705       .globl _main
706       .section .text
707       _main:
708           .size _main, . - _main
709
709       .text
710       .globl _main
711       .section .text
712       _main:
713           .size _main, . - _main
714
714       .text
715       .globl _main
716       .section .text
717       _main:
718           .size _main, . - _main
719
719       .text
720       .globl _main
721       .section .text
722       _main:
723           .size _main, . - _main
724
724       .text
725       .globl _main
726       .section .text
727       _main:
728           .size _main, . - _main
729
729       .text
730       .globl _main
731       .section .text
732       _main:
733           .size _main, . - _main
734
734       .text
735       .globl _main
736       .section .text
737       _main:
738           .size _main, . - _main
739
739       .text
740       .globl _main
741       .section .text
742       _main:
743           .size _main, . - _main
744
744       .text
745       .globl _main
746       .section .text
747       _main:
748           .size _main, . - _main
749
749       .text
750       .globl _main
751       .section .text
752       _main:
753           .size _main, . - _main
754
754       .text
755       .globl _main
756       .section .text
757       _main:
758           .size _main, . - _main
759
759       .text
760       .globl _main
761       .section .text
762       _main:
763           .size _main, . - _main
764
764       .text
765       .globl _main
766       .section .text
767       _main:
768           .size _main, . - _main
769
769       .text
770       .globl _main
771       .section .text
772       _main:
773           .size _main, . - _main
774
774       .text
775       .globl _main
776       .section .text
777       _main:
778           .size _main, . - _main
779
779       .text
780       .globl _main
781       .section .text
782       _main:
783           .size _main, . - _main
784
784       .text
785       .globl _main
786       .section .text
787       _main:
788           .size _main, . - _main
789
789       .text
790       .globl _main
791       .section .text
792       _main:
793           .size _main, . - _main
794
794       .text
795       .globl _main
796       .section .text
797       _main:
798           .size _main, . - _main
799
799       .text
800       .globl _main
801       .section .text
802       _main:
803           .size _main, . - _main
804
804       .text
805       .globl _main
806       .section .text
807       _main:
808           .size _main, . - _main
809
809       .text
810       .globl _main
811       .section .text
812       _main:
813           .size _main, . - _main
814
814       .text
815       .globl _main
816       .section .text
817       _main:
818           .size _main, . - _main
819
819       .text
820       .globl _main
821       .section .text
822       _main:
823           .size _main, . - _main
824
824       .text
825       .globl _main
826       .section .text
827       _main:
828           .size _main, . - _main
829
829       .text
830       .globl _main
831       .section .text
832       _main:
833           .size _main, . - _main
834
834       .text
835       .globl _main
836       .section .text
837       _main:
838           .size _main, . - _main
839
839       .text
840       .globl _main
841       .section .text
842       _main:
843           .size _main, . - _main
844
844       .text
845       .globl _main
846       .section .text
847       _main:
848           .size _main, . - _main
849
849       .text
850       .globl _main
851       .section .text
852       _main:
853           .size _main, . - _main
854
854       .text
855       .globl _main
856       .section .text
857       _main:
858           .size _main, . - _main
859
859       .text
860       .globl _main
861       .section .text
862       _main:
863           .size _main, . - _main
864
864       .text
865       .globl _main
866       .section .text
867       _main:
868           .size _main, . - _main
869
869       .text
870       .globl _main
871       .section .text
872       _main:
873           .size _main, . - _main
874
874       .text
875       .globl _main
876       .section .text
877       _main:
878           .size _main, . - _main
879
879       .text
880       .globl _main
881       .section .text
882       _main:
883           .size _main, . - _main
884
884       .text
885       .globl _main
886       .section .text
887       _main:
888           .size _main, . - _main
889
889       .text
890       .globl _main
891       .section .text
892       _main:
893           .size _main, . - _main
894
894       .text
895       .globl _main
896       .section .text
897       _main:
898           .size _main, . - _main
899
899       .text
900       .globl _main
901       .section .text
902       _main:
903           .size _main, . - _main
904
904       .text
905       .globl _main
906       .section .text
907       _main:
908           .size _main, . - _main
909
909       .text
910       .globl _main
911       .section .text
912       _main:
913           .size _main, . - _main
914
914       .text
915       .globl _main
916       .section .text
917       _main:
918           .size _main, . - _main
919
919       .text
920       .globl _main
921       .section .text
922       _main:
923           .size _main, . - _main
924
924       .text
925       .globl _main
926       .section .text
927       _main:
928           .size _main, . - _main
929
929       .text
930       .globl _main
931       .section .text
932       _main:
933           .size _main, . - _main
934
934       .text
935       .globl _main
936       .section .text
937       _main:
938           .size _main, . - _main
939
939       .text
940       .globl _main
941       .section .text
942       _main:
943           .size _main, . - _main
944
944       .text
945       .globl _main
946       .section .text
947       _main:
948           .size _main, . - _main
949
949       .text
950       .globl _main
951       .section .text
952       _main:
953           .size _main, . - _main
954
954       .text
955       .globl _main
956       .section .text
957       _main:
958           .size _main, . - _main
959
959       .text
960       .globl _main
961       .section .text
962       _main:
963           .size _main, . - _main
964
964       .text
965       .globl _main
966       .section .text
967       _main:
968           .size _main, . - _main
969
969       .text
970       .globl _main
971       .section .text
972       _main:
973           .size _main, . - _main
974
974       .text
975       .globl _main
976       .section .text
977       _main:
978           .size _main, . - _main
979
979       .text
980       .globl _main
981       .section .text
982       _main:
983           .size _main, . - _main
984
984       .text
985       .globl _main
986       .section .text
987       _main:
988           .size _main, . - _main
989
989       .text
990       .globl _main
991       .section .text
992       _main:
993           .size _main, . - _main
994
994       .text
995       .globl _main
996       .section .text
997       _main:
998           .size _main, . - _main
999
999       .text
1000      .globl _main
1001      .section .text
1002      _main:
1003          .size _main, . - _main
1004
1004       .text
1005       .globl _main
1006       .section .text
1007       _main:
1008           .size _main, . - _main
1009
1009       .text
1010      .globl _main
1011      .section .text
1012      _main:
1013          .size _main, . - _main
1014
1014       .text
1015       .globl _main
1016       .section .text
1017       _main:
1018           .size _main, . - _main
1019
1019       .text
1020       .globl _main
1021       .section .text
1022       _main:
1023           .size _main, . - _main
1024
1024       .text
1025       .globl _main
1026       .section .text
1027       _main:
1028           .size _main, . - _main
1029
1029       .text
1030       .globl _main
1031       .section .text
1032       _main:
1033           .size _main, . - _main
1034
1034       .text
1035       .globl _main
1036       .section .text
1037       _main:
1038           .size _main, . - _main
1039
1039       .text
1040       .globl _main
1041       .section .text
1042       _main:
1043           .size _main, . - _main
1044
1044       .text
1045       .globl _main
1046       .section .text
1047       _main:
1048           .size _main, . - _main
1049
1049       .text
1050       .globl _main
1051       .section .text
1052       _main:
1053           .size _main, . - _main
1054
1054       .text
1055       .globl _main
1056       .section .text
1057       _main:
1058           .size _main, . - _main
1059
1059       .text
1060       .globl _main
1061       .section .text
1062       _main:
1063           .size _main, . - _main
1064
1064       .text
1065       .globl _main
1066       .section .text
1067       _main:
1068           .size _main, . - _main
1069
1069       .text
1070       .globl _main
1071       .section .text
1072       _main:
1073           .size _main, . - _main
1074
1074       .text
1075       .globl _main
1076       .section .text
1077       _main:
1078           .size _main, . - _main
1079
1079       .text
1080       .globl _main
1081       .section .text
1082       _main:
1083           .size _main, . - _main
1084
1084       .text
1085       .globl _main
1086       .section .text
1087       _main:
1088           .size _main, . - _main
1089
1089       .text
1090       .globl _main
1091       .section .text
1092       _main:
1093           .size _main, . - _main
1094
1094       .text
1095       .globl _main
1096       .section .text
1097       _main:
1098           .size _main, . - _main
1099
1099       .text
1100      .globl _main
1101      .section .text
1102      _main:
1103          .size _main, . - _main
1104
1104       .text
1105       .globl _main
1106       .section .text
1107       _main:
1108           .size _main, . - _main
1109
1109       .text
1110      .globl _main
1111      .section .text
1112      _main:
1113          .size _main, . - _main
1114
1114       .text
1115       .globl _main
1116       .section .text
1117       _main:
1118           .size _main, . - _main
1119
1119       .text
1120       .globl _main
1121       .section .text
1122       _main:
1123           .size _main, . - _main
1124
1124       .text
1125       .globl _main
1126       .section .text
1127       _main:
1128           .size _main, . - _main
1129
1129       .text
1130       .globl _main
1131       .section .text
1132       _main:
1133           .size _main, . - _main
1134
1134       .text
1135       .globl _main
1136       .section .text
1137       _main:
1138           .size _main, . - _main
1139
1139       .text
1140       .globl _main
1141       .section .text
1142       _main:
1143           .size _main, . - _main
1144
1144       .text
1145       .globl _main
1146       .section .text
1147       _main:
1148           .size _main, . - _main
1149
1149       .text
1150       .globl _main
1151       .section .text
1152       _main:
1153           .size _main, . - _main
1154
1154       .text
1155       .globl _main
1156       .section .text
1157       _main:
1158           .size _main, . - _main
1159
1159       .text
1160       .globl _main
1161       .section .text
1162       _main:
1163           .size _main, . - _main
1164
1164       .text
1165       .globl _main
1166       .section .text
1167       _main:
1168           .size _main, . - _main
1169
1169       .text
1170       .globl _main
1171       .section .text
1172       _main:
1173           .size _main, . - _main
1174
1174       .text
1175       .globl _main
1176       .section .text
1177       _main:
1178           .size _main, . - _main
1179
1179       .text
1180       .globl _main
1181       .section .text
1182       _main:
1183           .size _main, . - _main
1184
1184       .text
1185       .globl _main
1186       .section .text
1187       _main:
1188           .size _main, . - _main
1189
1189       .text
1190       .globl _main
1191       .section .text
1192       _main:
1193           .size _main, . - _main
1194
1194       .text
1195       .globl _main
1196       .section .text
1197       _main:
1198           .size _main, . - _main
1199
1199       .text
1200       .globl _main
1201       .section .text
1202       _main:
1203           .size _main, . - _main
1204
1204       .text
1205       .globl _main
1206       .section .text
1207       _main:
1208           .size _main, . - _main
1209
1209       .text
1210       .globl _main
1211       .section .text
1212       _main:
1213           .size _main, . - _main
1214
1214       .text
1215       .globl _main
1216       .section .text
1217       _main:
1218           .size _main, . - _main
1219
1219       .text
1220       .globl _main
1221       .section .text
1222       _main:
1223           .size _main, . - _main
1224
1224       .text
1225       .globl _main
12
```

# MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there's a menu bar with 'File', 'Edit', 'Run', 'Help', 'Show/Hide Demo', and tabs for 'Interactive', 'Binary/Decimal', and 'Decimal/Binary'. Below the menu is a toolbar with buttons for 'Addition', 'Calculator', 'Bash', 'Loop', 'Stack Test', 'Hello World', 'Code Gen', 'Save String', 'Interactive', 'Binary/Decimal', and 'Decimal/Binary'. A 'Debug' button is also present.

The main area contains assembly code:

```
1 # Shows "Hello world!" at the top of the stack
2 .text
3 .globl _start
4 _start:
5    addi   $t0, $zero, 101 # a
6    addi   $t0, $zero, 101 # b
7    addi   $t0, $zero, 100 # 1
8    addi   $t0, $zero, 100 # 1
9    addi   $t0, $zero, 100 # 1
10   addi   $t0, $zero, 100 # 1
11   addi   $t0, $zero, 100 # 1
12   addi   $t0, $zero, 100 # 1
13   addi   $t0, $zero, 100 # 1
14   addi   $t0, $zero, 100 # 1
15   addi   $t0, $zero, 100 # 1
16   addi   $t0, $zero, 100 # 1
17   addi   $t0, $zero, 100 # 1
18   addi   $t0, $zero, 100 # 1
19   addi   $t0, $zero, 100 # 1
20   addi   $t0, $zero, 100 # 1
21   addi   $t0, $zero, 100 # 1
22   addi   $t0, $zero, 100 # d
23   addi   $t0, $zero, 100 # d
24   addi   $t0, $zero, 100 # d
25   addi   $t0, $zero, 100 # d
26   addi   $t0, $zero, 100 # d
27   addi   $t0, $zero, 100 # d
28   addi   $t0, $zero, 100 # d
29   addi   $t0, $zero, 100 # d
30   addi   $t0, $zero, 100 # d
31   addi   $t0, $zero, 100 # d
32   syscall
```

To the right of the code, there's a register table with columns for \$, T, A, V, Stack, and Log. The values are:

\$	T	A	V	Stack	Log
\$0	10				
\$1	9				
\$2	8				
\$3	7				
\$4	6				
\$5	5				
\$6	807				
\$7	418				

- **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

The screenshot shows the ShowMe Demo MIPS simulator. At the top, there's a menu bar with 'User', 'File', 'Run', 'Help', 'ShowMe Demo' (selected), 'Addition', 'Calculator', 'Itiva', 'Looper', 'Stack Test', 'Hello World', 'Code Gen Save String', 'Interactive', 'Binary Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with icons for 'Run', 'Break', 'Step', 'Stop', 'Reset', 'Run to cursor', 'Run to end', 'Run to exit', and 'Run to error'. To the right of the toolbar are links for 'User Guide', 'Unit Tests', and 'Docs'. The main area has tabs for 'Step' (selected) and 'Run' (disabled). A status bar at the bottom says 'Enable auto switching'. The assembly code window contains the following MIPS assembly code:

```
1 # Shows "Hello world!" at the top of the stack
2
3 .data
4 msg: .asciiz "Hello world!\n"
5
6 .text
7 addi $t0, $zero, 100 # $t0 = 100
8 addi $t1, $zero, 101 # $t1 = 101
9 addi $t2, $zero, 102 # $t2 = 102
10 addi $t3, $zero, 103 # $t3 = 103
11 addi $t4, $zero, 104 # $t4 = 104
12 addi $t5, $zero, 105 # $t5 = 105
13 addi $t6, $zero, 106 # $t6 = 106
14 addi $t7, $zero, 107 # $t7 = 107
15 addi $t8, $zero, 108 # $t8 = 108
16 addi $t9, $zero, 109 # $t9 = 109
17 addi $t10, $zero, 110 # $t10 = 110
18 addi $t11, $zero, 111 # $t11 = 111
19 addi $t12, $zero, 112 # $t12 = 112
20 addi $t13, $zero, 113 # $t13 = 113
21 addi $t14, $zero, 114 # $t14 = 114
22 addi $t15, $zero, 115 # $t15 = 115
23 addi $t16, $zero, 116 # $t16 = 116
24 addi $t17, $zero, 117 # $t17 = 117
25 addi $t18, $zero, 118 # $t18 = 118
26 addi $t19, $zero, 119 # $t19 = 119
27 addi $t20, $zero, 120 # $t20 = 120
28 addi $t21, $zero, 121 # $t21 = 121
29 addi $t22, $zero, 122 # $t22 = 122
30 addi $t23, $zero, 123 # $t23 = 123
31 addi $t24, $zero, 124 # $t24 = 124
32 addi $t25, $zero, 125 # $t25 = 125
33 addi $t26, $zero, 126 # $t26 = 126
34 addi $t27, $zero, 127 # $t27 = 127
35 addi $t28, $zero, 128 # $t28 = 128
36 addi $t29, $zero, 129 # $t29 = 129
37 addi $t30, $zero, 130 # $t30 = 130
38 addi $t31, $zero, 131 # $t31 = 131
39 addi $t32, $zero, 132 # $t32 = 132
40 addi $t33, $zero, 133 # $t33 = 133
41 addi $t34, $zero, 134 # $t34 = 134
42 addi $t35, $zero, 135 # $t35 = 135
43 addi $t36, $zero, 136 # $t36 = 136
44 addi $t37, $zero, 137 # $t37 = 137
45 addi $t38, $zero, 138 # $t38 = 138
46 addi $t39, $zero, 139 # $t39 = 139
47 addi $t40, $zero, 140 # $t40 = 140
48 addi $t41, $zero, 141 # $t41 = 141
49 addi $t42, $zero, 142 # $t42 = 142
50 addi $t43, $zero, 143 # $t43 = 143
51 addi $t44, $zero, 144 # $t44 = 144
52 addi $t45, $zero, 145 # $t45 = 145
53 addi $t46, $zero, 146 # $t46 = 146
54 addi $t47, $zero, 147 # $t47 = 147
55 addi $t48, $zero, 148 # $t48 = 148
56 addi $t49, $zero, 149 # $t49 = 149
57 addi $t50, $zero, 150 # $t50 = 150
58 addi $t51, $zero, 151 # $t51 = 151
59 addi $t52, $zero, 152 # $t52 = 152
60 addi $t53, $zero, 153 # $t53 = 153
61 addi $t54, $zero, 154 # $t54 = 154
62 addi $t55, $zero, 155 # $t55 = 155
63 addi $t56, $zero, 156 # $t56 = 156
64 addi $t57, $zero, 157 # $t57 = 157
65 addi $t58, $zero, 158 # $t58 = 158
66 addi $t59, $zero, 159 # $t59 = 159
67 addi $t60, $zero, 160 # $t60 = 160
68 addi $t61, $zero, 161 # $t61 = 161
69 addi $t62, $zero, 162 # $t62 = 162
70 addi $t63, $zero, 163 # $t63 = 163
71 addi $t64, $zero, 164 # $t64 = 164
72 addi $t65, $zero, 165 # $t65 = 165
73 addi $t66, $zero, 166 # $t66 = 166
74 addi $t67, $zero, 167 # $t67 = 167
75 addi $t68, $zero, 168 # $t68 = 168
76 addi $t69, $zero, 169 # $t69 = 169
77 addi $t70, $zero, 170 # $t70 = 170
78 addi $t71, $zero, 171 # $t71 = 171
79 addi $t72, $zero, 172 # $t72 = 172
80 addi $t73, $zero, 173 # $t73 = 173
81 addi $t74, $zero, 174 # $t74 = 174
82 addi $t75, $zero, 175 # $t75 = 175
83 addi $t76, $zero, 176 # $t76 = 176
84 addi $t77, $zero, 177 # $t77 = 177
85 addi $t78, $zero, 178 # $t78 = 178
86 addi $t79, $zero, 179 # $t79 = 179
87 addi $t80, $zero, 180 # $t80 = 180
88 addi $t81, $zero, 181 # $t81 = 181
89 addi $t82, $zero, 182 # $t82 = 182
90 addi $t83, $zero, 183 # $t83 = 183
91 addi $t84, $zero, 184 # $t84 = 184
92 addi $t85, $zero, 185 # $t85 = 185
93 addi $t86, $zero, 186 # $t86 = 186
94 addi $t87, $zero, 187 # $t87 = 187
95 addi $t88, $zero, 188 # $t88 = 188
96 addi $t89, $zero, 189 # $t89 = 189
97 addi $t90, $zero, 190 # $t90 = 190
98 addi $t91, $zero, 191 # $t91 = 191
99 addi $t92, $zero, 192 # $t92 = 192
100 addi $t93, $zero, 193 # $t93 = 193
101 addi $t94, $zero, 194 # $t94 = 194
102 addi $t95, $zero, 195 # $t95 = 195
103 addi $t96, $zero, 196 # $t96 = 196
104 addi $t97, $zero, 197 # $t97 = 197
105 addi $t98, $zero, 198 # $t98 = 198
106 addi $t99, $zero, 199 # $t99 = 199
107 addi $t100, $zero, 200 # $t100 = 200
108 addi $t101, $zero, 201 # $t101 = 201
109 addi $t102, $zero, 202 # $t102 = 202
110 addi $t103, $zero, 203 # $t103 = 203
111 addi $t104, $zero, 204 # $t104 = 204
112 addi $t105, $zero, 205 # $t105 = 205
113 addi $t106, $zero, 206 # $t106 = 206
114 addi $t107, $zero, 207 # $t107 = 207
115 addi $t108, $zero, 208 # $t108 = 208
116 addi $t109, $zero, 209 # $t109 = 209
117 addi $t110, $zero, 210 # $t110 = 210
118 addi $t111, $zero, 211 # $t111 = 211
119 addi $t112, $zero, 212 # $t112 = 212
120 addi $t113, $zero, 213 # $t113 = 213
121 addi $t114, $zero, 214 # $t114 = 214
122 addi $t115, $zero, 215 # $t115 = 215
123 addi $t116, $zero, 216 # $t116 = 216
124 addi $t117, $zero, 217 # $t117 = 217
125 addi $t118, $zero, 218 # $t118 = 218
126 addi $t119, $zero, 219 # $t119 = 219
127 addi $t120, $zero, 220 # $t120 = 220
128 addi $t121, $zero, 221 # $t121 = 221
129 addi $t122, $zero, 222 # $t122 = 222
130 addi $t123, $zero, 223 # $t123 = 223
131 addi $t124, $zero, 224 # $t124 = 224
132 addi $t125, $zero, 225 # $t125 = 225
133 addi $t126, $zero, 226 # $t126 = 226
134 addi $t127, $zero, 227 # $t127 = 227
135 addi $t128, $zero, 228 # $t128 = 228
136 addi $t129, $zero, 229 # $t129 = 229
137 addi $t130, $zero, 230 # $t130 = 230
138 addi $t131, $zero, 231 # $t131 = 231
139 addi $t132, $zero, 232 # $t132 = 232
140 addi $t133, $zero, 233 # $t133 = 233
141 addi $t134, $zero, 234 # $t134 = 234
142 addi $t135, $zero, 235 # $t135 = 235
143 addi $t136, $zero, 236 # $t136 = 236
144 addi $t137, $zero, 237 # $t137 = 237
145 addi $t138, $zero, 238 # $t138 = 238
146 addi $t139, $zero, 239 # $t139 = 239
147 addi $t140, $zero, 240 # $t140 = 240
148 addi $t141, $zero, 241 # $t141 = 241
149 addi $t142, $zero, 242 # $t142 = 242
150 addi $t143, $zero, 243 # $t143 = 243
151 addi $t144, $zero, 244 # $t144 = 244
152 addi $t145, $zero, 245 # $t145 = 245
153 addi $t146, $zero, 246 # $t146 = 246
154 addi $t147, $zero, 247 # $t147 = 247
155 addi $t148, $zero, 248 # $t148 = 248
156 addi $t149, $zero, 249 # $t149 = 249
157 addi $t150, $zero, 250 # $t150 = 250
158 addi $t151, $zero, 251 # $t151 = 251
159 addi $t152, $zero, 252 # $t152 = 252
160 addi $t153, $zero, 253 # $t153 = 253
161 addi $t154, $zero, 254 # $t154 = 254
162 addi $t155, $zero, 255 # $t155 = 255
163 addi $t156, $zero, 256 # $t156 = 256
164 addi $t157, $zero, 257 # $t157 = 257
165 addi $t158, $zero, 258 # $t158 = 258
166 addi $t159, $zero, 259 # $t159 = 259
167 addi $t160, $zero, 260 # $t160 = 260
168 addi $t161, $zero, 261 # $t161 = 261
169 addi $t162, $zero, 262 # $t162 = 262
170 addi $t163, $zero, 263 # $t163 = 263
171 addi $t164, $zero, 264 # $t164 = 264
172 addi $t165, $zero, 265 # $t165 = 265
173 addi $t166, $zero, 266 # $t166 = 266
174 addi $t167, $zero, 267 # $t167 = 267
175 addi $t168, $zero, 268 # $t168 = 268
176 addi $t169, $zero, 269 # $t169 = 269
177 addi $t170, $zero, 270 # $t170 = 270
178 addi $t171, $zero, 271 # $t171 = 271
179 addi $t172, $zero, 272 # $t172 = 272
180 addi $t173, $zero, 273 # $t173 = 273
181 addi $t174, $zero, 274 # $t174 = 274
182 addi $t175, $zero, 275 # $t175 = 275
183 addi $t176, $zero, 276 # $t176 = 276
184 addi $t177, $zero, 277 # $t177 = 277
185 addi $t178, $zero, 278 # $t178 = 278
186 addi $t179, $zero, 279 # $t179 = 279
187 addi $t180, $zero, 280 # $t180 = 280
188 addi $t181, $zero, 281 # $t181 = 281
189 addi $t182, $zero, 282 # $t182 = 282
190 addi $t183, $zero, 283 # $t183 = 283
191 addi $t184, $zero, 284 # $t184 = 284
192 addi $t185, $zero, 285 # $t185 = 285
193 addi $t186, $zero, 286 # $t186 = 286
194 addi $t187, $zero, 287 # $t187 = 287
195 addi $t188, $zero, 288 # $t188 = 288
196 addi $t189, $zero, 289 # $t189 = 289
197 addi $t190, $zero, 290 # $t190 = 290
198 addi $t191, $zero, 291 # $t191 = 291
199 addi $t192, $zero, 292 # $t192 = 292
200 addi $t193, $zero, 293 # $t193 = 293
201 addi $t194, $zero, 294 # $t194 = 294
202 addi $t195, $zero, 295 # $t195 = 295
203 addi $t196, $zero, 296 # $t196 = 296
204 addi $t197, $zero, 297 # $t197 = 297
205 addi $t198, $zero, 298 # $t198 = 298
206 addi $t199, $zero, 299 # $t199 = 299
207 addi $t200, $zero, 300 # $t200 = 300
208 addi $t201, $zero, 301 # $t201 = 301
209 addi $t202, $zero, 302 # $t202 = 302
210 addi $t203, $zero, 303 # $t203 = 303
211 addi $t204, $zero, 304 # $t204 = 304
212 addi $t205, $zero, 305 # $t205 = 305
213 addi $t206, $zero, 306 # $t206 = 306
214 addi $t207, $zero, 307 # $t207 = 307
215 addi $t208, $zero, 308 # $t208 = 308
216 addi $t209, $zero, 309 # $t209 = 309
217 addi $t210, $zero, 310 # $t210 = 310
218 addi $t211, $zero, 311 # $t211 = 311
219 addi $t212, $zero, 312 # $t212 = 312
220 addi $t213, $zero, 313 # $t213 = 313
221 addi $t214, $zero, 314 # $t214 = 314
222 addi $t215, $zero, 315 # $t215 = 315
223 addi $t216, $zero, 316 # $t216 = 316
224 addi $t217, $zero, 317 # $t217 = 317
225 addi $t218, $zero, 318 # $t218 = 318
226 addi $t219, $zero, 319 # $t219 = 319
227 addi $t220, $zero, 320 # $t220 = 320
228 addi $t221, $zero, 321 # $t221 = 321
229 addi $t222, $zero, 322 # $t222 = 322
230 addi $t223, $zero, 323 # $t223 = 323
231 addi $t224, $zero, 324 # $t224 = 324
232 addi $t225, $zero, 325 # $t225 = 325
233 addi $t226, $zero, 326 # $t226 = 326
234 addi $t227, $zero, 327 # $t227 = 327
235 addi $t228, $zero, 328 # $t228 = 328
236 addi $t229, $zero, 329 # $t229 = 329
237 addi $t230, $zero, 330 # $t230 = 330
238 addi $t231, $zero, 331 # $t231 = 331
239 addi $t232, $zero, 332 # $t232 = 332
240 addi $t233, $zero, 333 # $t233 = 333
241 addi $t234, $zero, 334 # $t234 = 334
242 addi $t235, $zero, 335 # $t235 = 335
243 addi $t236, $zero, 336 # $t236 = 336
244 addi $t237, $zero, 337 # $t237 = 337
245 addi $t238, $zero, 338 # $t238 = 338
246 addi $t239, $zero, 339 # $t239 = 339
247 addi $t240, $zero, 340 # $t240 = 340
248 addi $t241, $zero, 341 # $t241 = 341
249 addi $t242, $zero, 342 # $t242 = 342
250 addi $t243, $zero, 343 # $t243 = 343
251 addi $t244, $zero, 344 # $t244 = 344
252 addi $t245, $zero, 345 # $t245 = 345
253 addi $t246, $zero, 346 # $t246 = 346
254 addi $t247, $zero, 347 # $t247 = 347
255 addi $t248, $zero, 348 # $t248 = 348
256 addi $t249, $zero, 349 # $t249 = 349
257 addi $t250, $zero, 350 # $t250 = 350
258 addi $t251, $zero, 351 # $t251 = 351
259 addi $t252, $zero, 352 # $t252 = 352
260 addi $t253, $zero, 353 # $t253 = 353
261 addi $t254, $zero, 354 # $t254 = 354
262 addi $t255, $zero, 355 # $t255 = 355
263 addi $t256, $zero, 356 # $t256 = 356
264 addi $t257, $zero, 357 # $t257 = 357
265 addi $t258, $zero, 358 # $t258 = 358
266 addi $t259, $zero, 359 # $t259 = 359
267 addi $t260, $zero, 360 # $t260 = 360
268 addi $t261, $zero, 361 # $t261 = 361
269 addi $t262, $zero, 362 # $t262 = 362
270 addi $t263, $zero, 363 # $t263 = 363
271 addi $t264, $zero, 364 # $t264 = 364
272 addi $t265, $zero, 365 # $t265 = 365
273 addi $t266, $zero, 366 # $t266 = 366
274 addi $t267, $zero, 367 # $t267 = 367
275 addi $t268, $zero, 368 # $t268 = 368
276 addi $t269, $zero, 369 # $t269 = 369
277 addi $t270, $zero, 370 # $t270 = 370
278 addi $t271, $zero, 371 # $t271 = 371
279 addi $t272, $zero, 372 # $t272 = 372
280 addi $t273, $zero, 373 # $t273 = 373
281 addi $t274, $zero, 374 # $t274 = 374
282 addi $t275, $zero, 375 # $t275 = 375
283 addi $t276, $zero, 376 # $t276 = 376
284 addi $t277, $zero, 377 # $t277 = 377
285 addi $t278, $zero, 378 # $t278 = 378
286 addi $t279, $zero, 379 # $t279 = 379
287 addi $t280, $zero, 380 # $t280 = 380
288 addi $t281, $zero, 381 # $t281 = 381
289 addi $t282, $zero, 382 # $t282 = 382
290 addi $t283, $zero, 383 # $t283 = 383
291 addi $t284, $zero, 384 # $t284 = 384
292 addi $t285, $zero, 385 # $t285 = 385
293 addi $t286, $zero, 386 # $t286 = 386
294 addi $t287, $zero, 387 # $t287 = 387
295 addi $t288, $zero, 388 # $t288 = 388
296 addi $t289, $zero, 389 # $t289 = 389
297 addi $t290, $zero, 390 # $t290 = 390
298 addi $t291, $zero, 391 # $t291 = 391
299 addi $t292, $zero, 392 # $t292 = 392
300 addi $t293, $zero, 393 # $t293 = 393
301 addi $t294, $zero, 394 # $t294 = 394
302 addi $t295, $zero, 395 # $t295 = 395
303 addi $t296, $zero, 396 # $t296 = 396
304 addi $t297, $zero, 397 # $t297 = 397
305 addi $t298, $zero, 398 # $t298 = 398
306 addi $t299, $zero, 399 # $t299 = 399
307 addi $t300, $zero, 400 # $t300 = 400
308 addi $t301, $zero, 401 # $t301 = 401
309 addi $t302, $zero, 402 # $t302 = 402
310 addi $t303, $zero, 403 # $t303 = 403
311 addi $t304, $zero, 404 # $t304 = 404
312 addi $t305, $zero, 405 # $t305 = 405
313 addi $t306, $zero, 406 # $t306 = 406
314 addi $t307, $zero, 407 # $t307 = 407
315 addi $t308, $zero, 408 # $t308 = 408
316 addi $t309, $zero, 409 # $t309 = 409
317 addi $t310, $zero, 410 # $t310 = 410
318 addi $t311, $zero, 411 # $t311 = 411
319 addi $t312, $zero, 412 # $t312 = 412
320 addi $t313, $zero, 413 # $t313 = 413
321 addi $t314, $zero, 414 # $t314 = 414
322 addi $t315, $zero, 415 # $t315 = 415
323 addi $t316, $zero, 416 # $t316 = 416
324 addi $t317, $zero, 417 # $t317 = 417
325 addi $t318, $zero, 418 # $t318 = 418
326 addi $t319, $zero, 419 # $t319 = 419
327 addi $t320, $zero, 420 # $t320 = 420
328 addi $t321, $zero, 421 # $t321 = 421
329 addi $t322, $zero, 422 # $t322 = 422
330 addi $t323, $zero, 423 # $t323 = 423
331 addi $t324, $zero, 424 # $t324 = 424
332 addi $t325, $zero, 425 # $t325 = 425
333 addi $t326, $zero, 426 # $t326 = 426
334 addi $t327, $zero, 427 # $t327 = 427
335 addi $t328, $zero, 428 # $t328 = 428
336 addi $t329, $zero, 429 # $t329 = 429
337 addi $t330, $zero, 430 # $t330 = 430
338 addi $t331, $zero, 431 # $t331 = 431
339 addi $t332, $zero, 432 # $t332 = 432
340 addi $t333, $zero, 433 # $t333 = 433
341 addi $t334, $zero, 434 # $t334 = 434
342 addi $t335, $zero, 435 # $t335 = 435
343 addi $t336, $zero, 436 # $t336 = 436
344 addi $t337, $zero, 437 # $t337 = 437
345 addi $t338, $zero, 438 # $t338 = 438
346 addi $t339, $zero, 439 # $t339 = 439
347 addi $t340, $zero, 440 # $t340 = 440
348 addi $t341, $zero, 441 # $t341 = 441
349 addi $t342, $zero, 442 # $t342 = 442
350 addi $t343, $zero, 443 # $t343 = 443
351 addi $t344, $zero, 444 # $t344 = 444
352 addi $t345, $zero, 445 # $t345 = 445
353 addi $t346, $zero, 446 # $t346 = 446
354 addi $t347, $zero, 447 # $t347 = 447
355 addi $t348, $zero, 448 # $t348 = 448
356 addi $t349, $zero, 449 # $t349 = 449
357 addi $t350, $zero, 450 # $t350 = 450
358 addi $t351, $zero, 451 # $t351 = 451
359 addi $t352, $zero, 452 # $t352 = 452
360 addi $t353, $zero, 453 # $t353 = 453
361 addi $t354, $zero, 454 # $t354 = 454
362 addi $t355, $zero, 455 # $t355 = 455
363 addi $t356, $zero, 456 # $t356 = 456
364 addi $t357, $zero, 457 # $t357 = 457
365 addi $t358, $zero, 458 # $t358 = 458
366 addi $t359, $zero, 459 # $t359 = 459
367 addi $t360, $zero, 460 # $t360 = 460
368 addi $t361, $zero, 461 # $t361 = 461
369 addi $t362, $zero, 462 # $t362 = 462
370 addi $t363, $zero, 463 # $t363 = 463
371 addi $t364, $zero, 464 # $t364 = 464
372 addi $t365, $zero, 465 # $t365 = 465
373 addi $t366, $zero, 466 # $t366 = 466
374 addi $t367, $zero, 467 # $t367 = 467
375 addi $t368, $zero, 468 # $t368 = 468
376 addi $t369, $zero, 469 # $t369 = 469
377 addi $t370, $zero, 470 # $t370 = 470
378 addi $t371, $zero, 471 # $t371 = 471
379 addi $t372, $zero, 472 # $t372 = 472
380 addi $t373, $zero, 473 # $t373 = 473
381 addi $t374, $zero, 474 # $t374 = 474
382 addi $t375, $zero, 475 # $t375 = 475
383 addi $t376, $zero, 476 # $t376 = 476
384 addi $t377, $zero, 477 # $t377 = 477
385 addi $t378, $zero, 478 # $t378 = 478
386 addi $t379, $zero, 479 # $t379 = 479
387 addi $t380, $zero, 480 # $t380 = 480
388 addi $t381, $zero, 481 # $t381 = 481
389 addi $t382, $zero, 482 # $t382 = 482
390 addi $t383, $zero, 483 # $t383 = 483
391 addi $t384, $zero, 484 # $t384 = 484
392 addi $t385, $zero, 485 # $t385 = 485
393 addi $t386, $zero, 486 # $t386 = 486
394 addi $t387, $zero, 487 # $t387 = 487
395 addi $t388, $zero, 488 # $t388 = 488
396 addi $t389, $zero, 489 # $t389 = 489
397 addi $t390, $zero, 490 # $t390 = 490
398 addi $t391, $zero, 491 # $t391 = 491
399 addi $t392, $zero, 492 # $t392 = 492
400 addi $t393, $zero, 493 # $t393 = 493
401 addi $t394, $zero, 494 # $t394 = 494
402 addi $t395, $zero, 495 # $t395 = 495
403 addi $t396, $zero, 496 # $t396 = 496
404 addi $t397, $zero, 497 # $t397 = 497
405 addi $t398, $zero, 498 # $t398 = 498
406 addi $t399, $zero, 499 # $t399 = 499
407 addi $t400, $zero, 500 # $t400 = 500
408 addi $t401, $zero, 501 # $t401 = 501
409 addi $t402, $zero, 502 # $t402 = 502
410 addi $t403, $zero, 503 # $t403 = 503
411 addi $t404, $zero, 504 # $t404 = 504
412 addi $t405, $zero, 505 # $t405 = 505
413 addi $t406, $zero, 506 # $t406 = 506
414 addi $t407, $zero, 507 # $t407 = 507
415 addi $t408, $zero, 508 # $t408 = 508
416 addi $t409, $zero, 509 # $t409 = 509
417 addi $t410, $zero, 510 # $t410 = 510
418 addi $t411, $zero, 511 # $t411 = 511
419 addi $t412, $zero, 512 # $
```

# MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there's a menu bar with 'File', 'Edit', 'Run', 'Help', 'Show/Hide Demo', and tabs for 'Addition', 'Bubble Sort', 'Interactive', 'Binary/Decimal', and 'Decimal/Binary'. Below the menu is a toolbar with buttons for 'Code Gen Save String', 'Interactive', 'Binary/Decimal', and 'Decimal/Binary'. A 'Debug' button is also present. On the right, there are links for 'User Guide', 'Unit Tests', and 'Docs'. The main area contains assembly code and a register dump.

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

# MIPS Commands

The screenshot shows the ShowMe Demo MIPS simulator. At the top, there are tabs for User, ShowMe Demo, and Run. Below the tabs are buttons for Addition, Counter, If-Else, Loops, Stack Test, Hello World, Code Gen, Save String, Interactive, Binary/Decimal, Decimal/Binary, and Debug. The main area displays assembly code and a register dump.

**Assembly Code:**

```
1 # Shows "Hello world" at the top of the stack
2 .text
3 .globl _start
4 _start:
5    addi   $s0, $zero, 101 # $s0 = 101
6    addi   $s1, $zero, 102 # $s1 = 102
7    addi   $s2, $zero, 103 # $s2 = 103
8    addi   $s3, $zero, 104 # $s3 = 104
9    addi   $s4, $zero, 105 # $s4 = 105
10   addi   $s5, $zero, 106 # $s5 = 106
11   addi   $s6, $zero, 107 # $s6 = 107
12   addi   $s7, $zero, 108 # $s7 = 108
13   addi   $s8, $zero, 109 # $s8 = 109
14   addi   $s9, $zero, 110 # $s9 = 110
15   addi   $s10, $zero, 111 # $s10 = 111
16   addi   $s11, $zero, 112 # $s11 = 112
17   addi   $s12, $zero, 113 # $s12 = 113
18   addi   $s13, $zero, 114 # $s13 = 114
19   addi   $s14, $zero, 115 # $s14 = 115
20   addi   $s15, $zero, 116 # $s15 = 116
21   addi   $s16, $zero, 117 # $s16 = 117
22   addi   $s17, $zero, 118 # $s17 = 118
23   addi   $s18, $zero, 119 # $s18 = 119
24   addi   $s19, $zero, 120 # $s19 = 120
25   addi   $s20, $zero, 121 # $s20 = 121
26   addi   $s21, $zero, 122 # $s21 = 122
27   addi   $s22, $zero, 0 # (null)
28
29   addi   $s23, $s0, 4 # $s23 = 101
30
31   addi   $s24, $s23, 0x40 # print to the log
32   syscall
```

**Register Dump:**

S	T	A	V	Stack	Log
s0				101	
s1				102	
s2				103	
s3				104	
s4				105	
s5				106	
s6				107	
s7				108	
s8				109	
s9				110	
s10				111	
s11				112	
s12				113	
s13				114	
s14				115	
s15				116	
s16				117	
s17				118	
s18				119	
s19				120	
s20				121	
s21				122	
s22				0	(null)
s23				101	
s24				40	# print to the log
s25				418	

- **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 are tabs for 'Show/Hide Demo', 'User Guide', 'Unit Tests', and 'Doc'. Below the tabs are several buttons: 'Addition Counter', 'Btav', 'Looper', 'Stack Test', 'Hello World', 'Code Gen Save String', 'Interactive', 'Binary/Decimal', 'Decimal/Binary', and 'Debug'. The main area contains assembly code and a register dump.

**Assembly Code:**

```
# Shows "Hello world" at the top of the stack
1    .data
2    msg: .asciiz "Hello world"
3
4    .text
5    .globl _start
6
7    _start:
8        addi $t0, $zero, 100 # $t0 = 100
9        addi $t1, $zero, 100 # $t1 = 100
10       addi $t2, $zero, 100 # $t2 = 100
11       addi $t3, $zero, 100 # $t3 = 100
12       addi $t4, $zero, 100 # $t4 = 100
13       addi $t5, $zero, 100 # $t5 = 100
14       addi $t6, $zero, 100 # $t6 = 100
15       addi $t7, $zero, 100 # $t7 = 100
16       addi $t8, $zero, 100 # $t8 = 100
17       addi $t9, $zero, 100 # $t9 = 100
18       addi $t10, $zero, 100 # $t10 = 100
19       addi $t11, $zero, 100 # $t11 = 100
20       addi $t12, $zero, 100 # $t12 = 100
21       addi $t13, $zero, 100 # $t13 = 100
22       addi $t14, $zero, 100 # $t14 = 100
23       addi $t15, $zero, 100 # $t15 = 100
24       addi $t16, $zero, 100 # $t16 = 100
25       addi $t17, $zero, 100 # $t17 = 100
26       addi $t18, $zero, 100 # $t18 = 100
27       addi $t19, $zero, 100 # $t19 = 100
28       addi $t20, $zero, 100 # $t20 = 100
29       addi $t21, $zero, 100 # $t21 = 100
30       addi $t22, $zero, 100 # $t22 = 100
31       addi $t23, $zero, 100 # $t23 = 100
32       addi $t24, $zero, 100 # $t24 = 100
33       addi $t25, $zero, 100 # $t25 = 100
34       addi $t26, $zero, 100 # $t26 = 100
35       addi $t27, $zero, 100 # $t27 = 100
36       addi $t28, $zero, 100 # $t28 = 100
37       addi $t29, $zero, 100 # $t29 = 100
38       addi $t30, $zero, 100 # $t30 = 100
39       addi $t31, $zero, 100 # $t31 = 100
40
41       li $v0, 4 # $v0 = 4 for print string
42       la $a0, msg # $a0 = address of msg
43       syscall
44
45       li $v0, 10 # print to the log
46       syscall
```

**Registers:**

S	T	A	V	Stack	Log
\$0				10	
\$1				9	
\$2				8	
\$3				7	
\$4				6	
\$5				5	
\$6				4	
\$7				3	
\$8				2	
\$9				1	
\$10				0	
\$11				100	
\$12				99	
\$13				98	
\$14				97	
\$15				96	
\$16				95	
\$17				94	
\$18				93	
\$19				92	
\$20				91	
\$21				90	
\$22				89	
\$23				88	
\$24				87	
\$25				86	
\$26				85	
\$27				84	
\$28				83	
\$29				82	
\$30				81	
\$31				80	

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

# MIPS Commands

The screenshot shows the ShowMe Demo interface. At the top, there are tabs for User, ShowMe Demo, Addition, Subtraction, Bitwise, Looper, Stack Test, Hello World, Code Gen, Save String, Interactive, Binary, Decimal, and Debug. The Debug tab is selected. Below the tabs is a text area containing MIPS assembly code. To the right is a table showing register values.

S	T	A	V	Stack	Log
\$0	10				
\$1	9				
\$2	8				
\$3	22				
\$4	60				
\$5	61				
\$6	807				
\$7	418				

Assembly code (partial):

```
1 # Shows "Hello world" at the top of the stack
2 .text
3 .globl _start
4 _start:
5    addi   $t0, $zero, 100 # 1
6    addi   $t1, $zero, 100 # 1
7    addi   $t2, $zero, 100 # 1
8    addi   $t3, $zero, 100 # 1
9    addi   $t4, $zero, 100 # 1
10   addi   $t5, $zero, 100 # 1
11   addi   $t6, $zero, 100 # 1
12   addi   $t7, $zero, 100 # 1
13   addi   $t8, $zero, 100 # 1
14   addi   $t9, $zero, 100 # 1
15   addi   $t10, $zero, 100 # 1
16   addi   $t11, $zero, 100 # 1
17   addi   $t12, $zero, 100 # 1
18   addi   $t13, $zero, 100 # 1
19   addi   $t14, $zero, 100 # 1
20   addi   $t15, $zero, 100 # 1
21   addi   $t16, $zero, 100 # 1
22   addi   $t17, $zero, 100 # 1
23   addi   $t18, $zero, 100 # 1
24   addi   $t19, $zero, 100 # 1
25   addi   $t20, $zero, 100 # 1
26   addi   $t21, $zero, 100 # 1
27   addi   $t22, $zero, 0 # (all)
28
29   addi   $t23, $zero, 4 # 4 is for print string
30   addi   $t24, $zero, 0 # print to the log
31   syscall
```

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

## MIPS Commands

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

## MIPS Commands

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

# Challenge Problem:

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

```
# Store 'Hello world!' at the top of the stack
ADDI $t0, zero, 72 #B
#H $t0, 0($sp)
ADD $t0, zero, 151 #e
#H $t0, 1($sp)
ADD $t0, zero, 150 #d
#H $t0, 2($sp)
ADD $t0, zero, 149 #c
#H $t0, 3($sp)
ADD $t0, zero, 111 #o
#H $t0, 4($sp)
ADD $t0, zero, 32 #($spave)
#H $t0, 5($sp)
ADD $t0, zero, 113 #w
#H $t0, 6($sp)
ADD $t0, zero, 111 #o
#H $t0, 7($sp)
ADD $t0, zero, 114 #r
#H $t0, 8($sp)
ADD $t0, zero, 108 #l
#H $t0, 9($sp)
ADD $t0, zero, 109 #d
#H $t0, 10($sp)
ADD $t0, zero, 33 #i
#H $t0, 11($sp)
ADD $t0, zero, 0 #($null)
#H $t0, 12($sp)
#H $t0, 0($sp)

ADDI $t0, zero, 4 # 4 is for print string
#H $t0, 0($sp)
#H $t0, 0($sp) # point to the log
#H $t0, 0($sp)
```

## (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

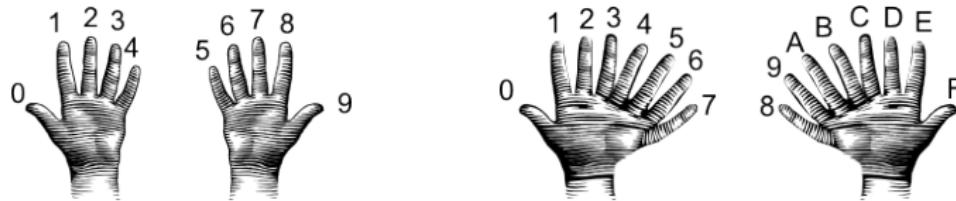
## (Demo with WeMIPS)

# Today's Topics



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

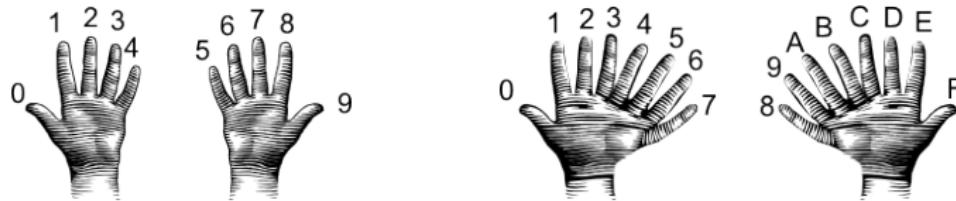
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

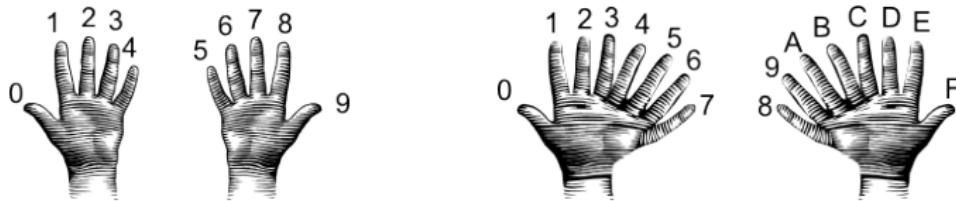
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

# Hexadecimal to Decimal: Converting Between Bases

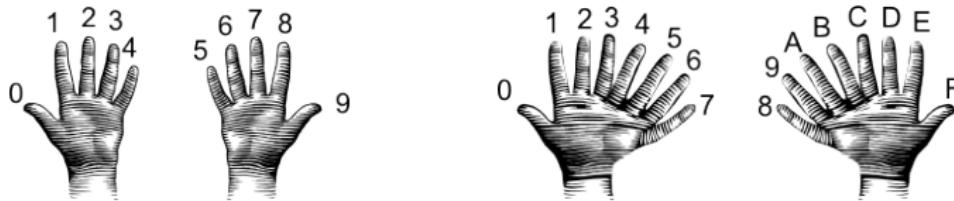


(from i-programmer.info)

- From hexadecimal to decimal:

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

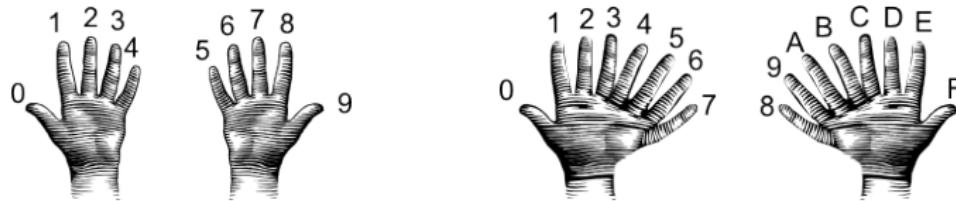
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

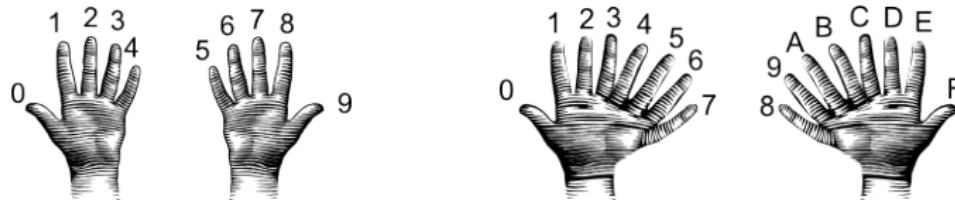
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:
  - Convert first digit to decimal and multiple by 16.
  - Convert second digit to decimal and add to total.
  - Example: what is 2A as a decimal number?  
2 in decimal is 2.  $2 \times 16$  is 32.

# Hexadecimal to Decimal: Converting Between Bases

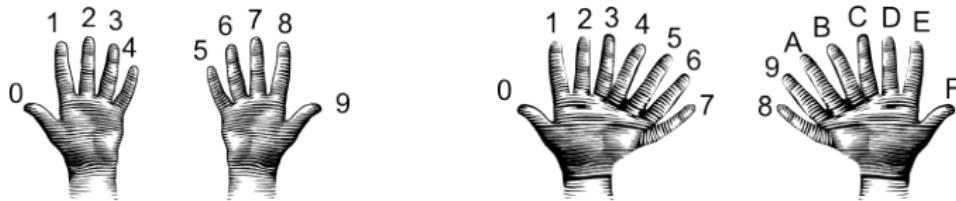


(from i-programmer.info)

- From hexadecimal to decimal:

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

# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal:

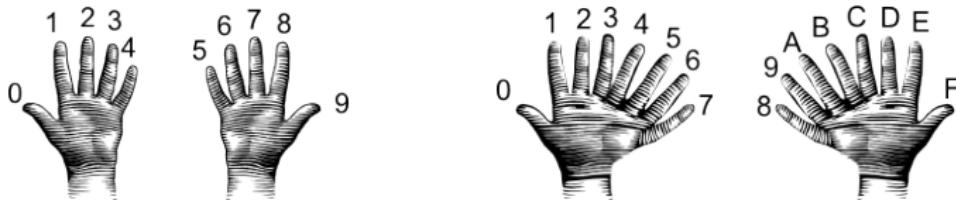
- Convert first digit to decimal and multiple by 16.
- Convert second digit to decimal and add to total.
- Example: what is 2A as a decimal number?

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

- 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.

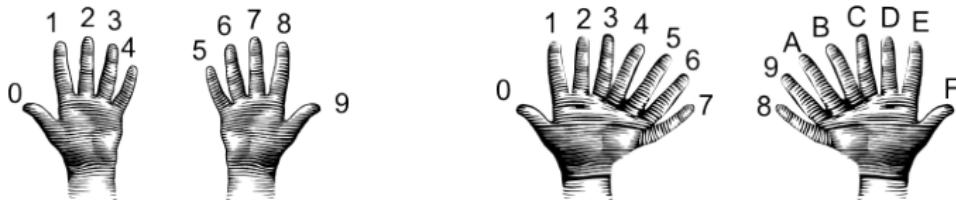
A in decimal digits is 10.

$32 + 10$  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:

- 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.

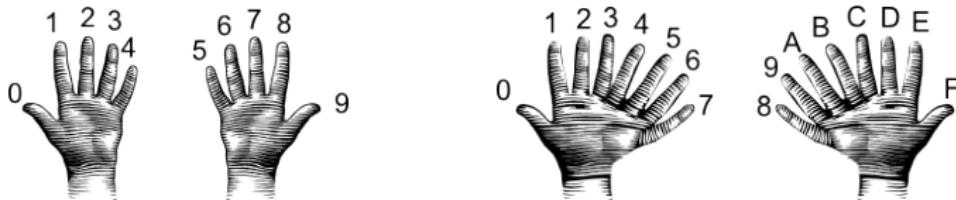
$32 + 10$  is 42.

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:

- 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.

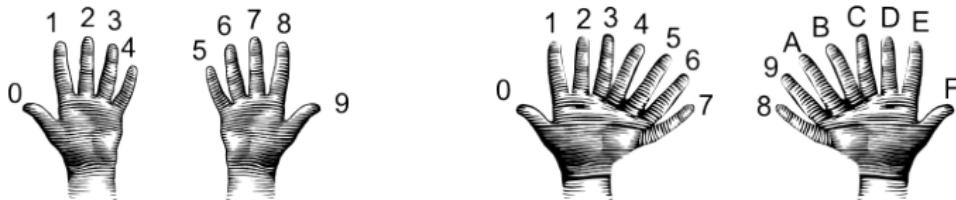
$32 + 10$  is 42.

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:

- 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.

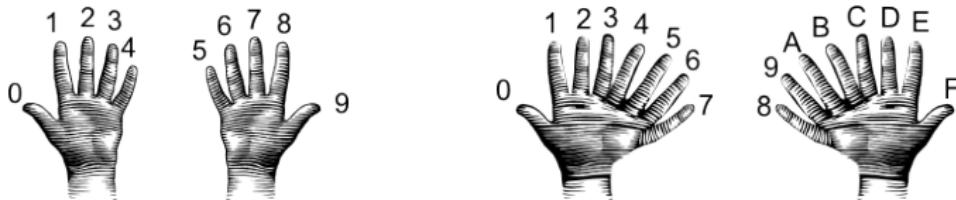
Answer is 42.

- 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:

- 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.

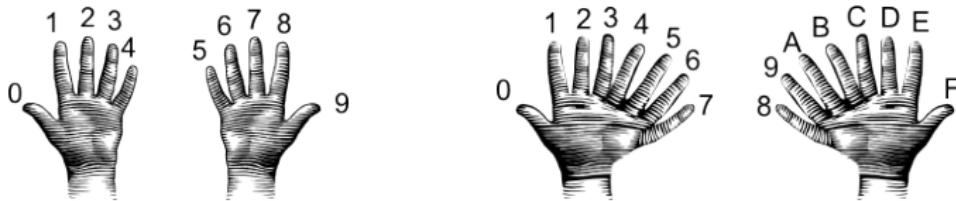
- Example: what is 99 as a decimal number?

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:

- 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?

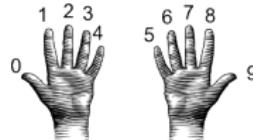
9 in decimal is 9.  $9 \times 16$  is 144.

9 in decimal digits is 9

$144 + 9$  is 153.

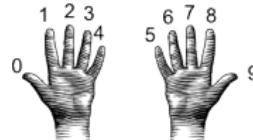
Answer is 153.

# Decimal to Binary: Converting Between Bases



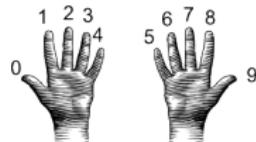
- From decimal to binary:
  - Divide by 128 ( $= 2^7$ ). Quotient is the first digit.

# Decimal to Binary: Converting Between Bases



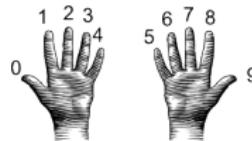
- From decimal to binary:
  - ▶ Divide by  $128 (= 2^7)$ . Quotient is the first digit.
  - ▶ Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.

# Decimal to Binary: Converting Between Bases



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

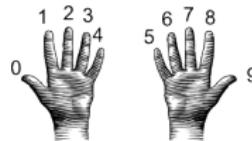
# Decimal to Binary: Converting Between Bases



- 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.

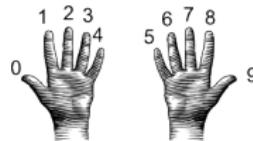
# Decimal to Binary: Converting Between Bases



- 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.

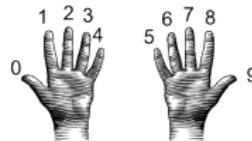
# Decimal to Binary: Converting Between Bases



- 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.

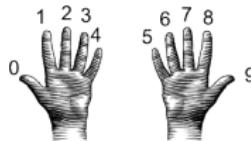
# Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

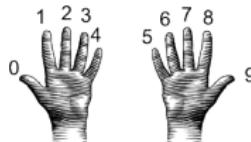
# Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

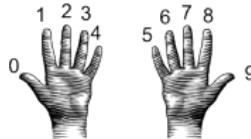
# Decimal to Binary: Converting Between Bases



- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

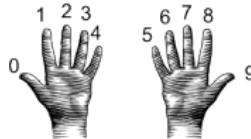


- From decimal to binary:

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

130/128 is 1 rem 2.

# Decimal to Binary: Converting Between Bases

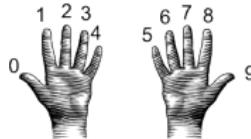


- From decimal to binary:

- ▶ 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



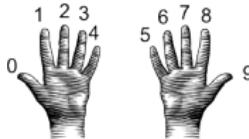
- From decimal to binary:

- ▶ 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



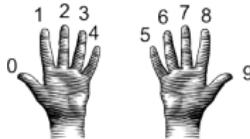
- From decimal to binary:

- ▶ 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



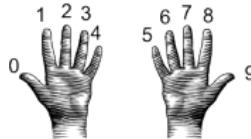
- From decimal to binary:

- 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



- From decimal to binary:

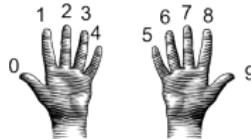
- ▶ Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- ▶ Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- ▶ Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- ▶ Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- ▶ Divide remainder by  $8 (= 2^3)$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

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

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

2/32 is 0 rem 2.

# Decimal to Binary: Converting Between Bases



- From decimal to binary:

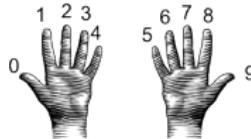
- ▶ Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- ▶ Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- ▶ Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- ▶ Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- ▶ Divide remainder by  $8 (= 2^3)$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

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

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

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

# Decimal to Binary: Converting Between Bases



- From decimal to binary:

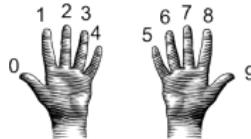
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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



- 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?

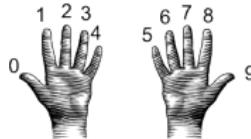
130/128 is 1 rem 2. First digit is 1: 1...

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



- 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?

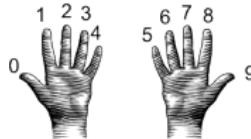
130/128 is 1 rem 2. First digit is 1: 1...

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



- 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?

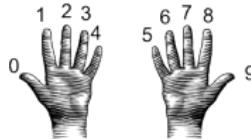
130/128 is 1 rem 2. First digit is 1: 1...

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



- 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...

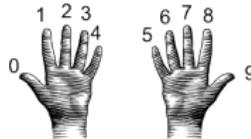
2/64 is 0 rem 2. Next digit is 0: 10...

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



- 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...

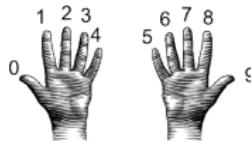
2/64 is 0 rem 2. Next digit is 0: 10...

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



- 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...

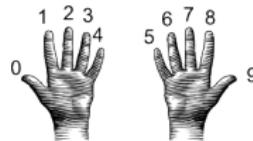
2/64 is 0 rem 2. Next digit is 0: 10...

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



- 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...

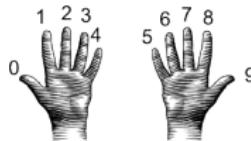
2/32 is 0 rem 2. Next digit is 0: 100...

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



- 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...

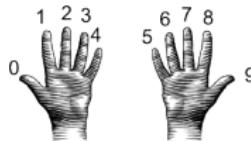
2/32 is 0 rem 2. Next digit is 0: 100...

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



- 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...

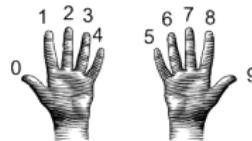
2/32 is 0 rem 2. Next digit is 0: 100...

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



- 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...

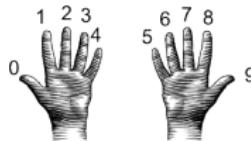
2/16 is 0 rem 2. Next digit is 0: 1000...

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



- 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...

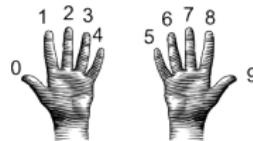
2/16 is 0 rem 2. Next digit is 0: 1000...

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



- 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...

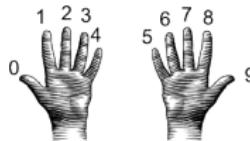
2/16 is 0 rem 2. Next digit is 0: 1000...

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



- 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...

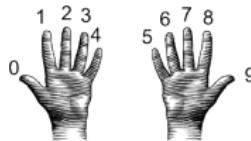
2/8 is 0 rem 2. Next digit is 0: 10000...

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...

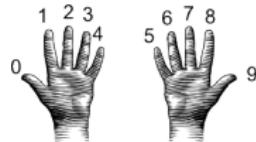
2/4 is 0 remainder 2. Next digit is 0: 100000...

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

Adding the last remainder:

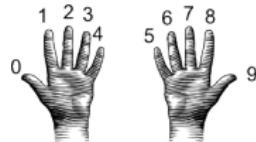
10000010

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

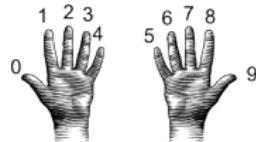
# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

$99/128$  is 0 rem 99.

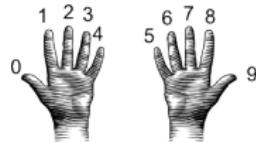
# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

# Decimal to Binary: Converting Between Bases

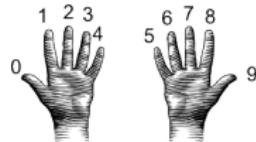


- Example: what is 99 in binary notation?

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

99/64 is 1 rem 35.

# Decimal to Binary: Converting Between Bases

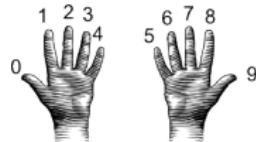


- Example: what is 99 in binary notation?

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

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

# Decimal to Binary: Converting Between Bases

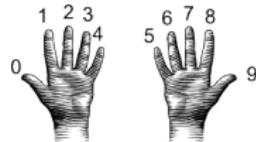


- Example: what is 99 in binary notation?

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

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

# Decimal to Binary: Converting Between Bases



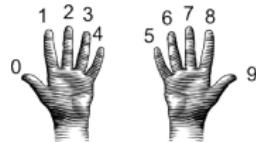
- Example: what is 99 in binary notation?

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

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

35/32 is 1 rem 3.

# Decimal to Binary: Converting Between Bases



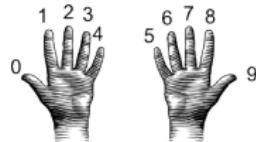
- Example: what is 99 in binary notation?

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

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

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

# Decimal to Binary: Converting Between Bases



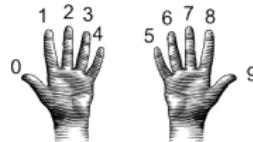
- Example: what is 99 in binary notation?

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

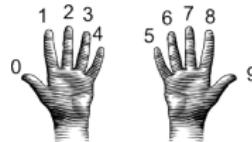
99/128 is 0 rem 99. First digit is 0: 0...

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

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

3/16 is 0 rem 3.

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

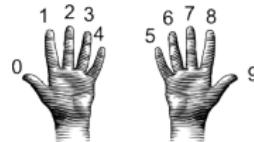
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

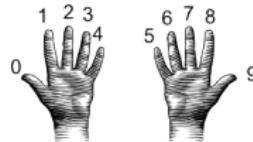
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

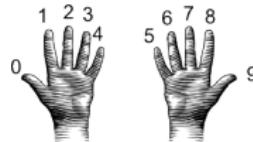
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

3/8 is 0 rem 3.

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

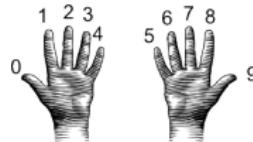
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

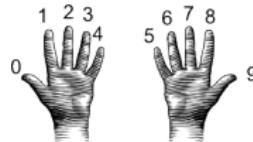
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

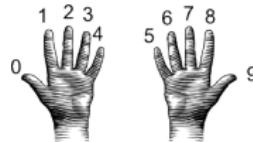
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

3/4 is 0 remainder 3.

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

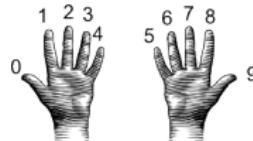
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

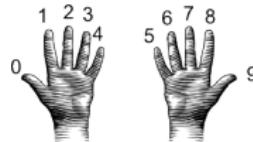
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

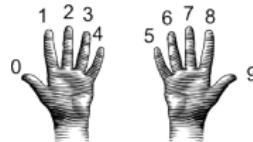
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1.

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

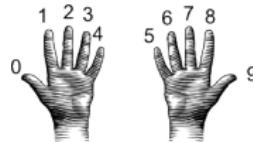
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

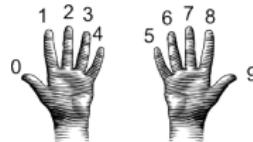
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

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

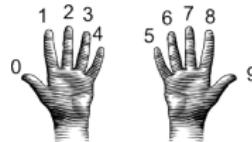
3/8 is 0 rem 3. Next digit is 0: 01100...

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

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

Adding the last remainder: 01100011

# Decimal to Binary: Converting Between Bases



- Example: what is 99 in binary notation?

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

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

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

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

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

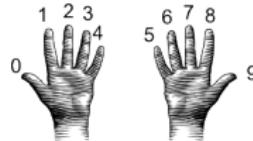
3/4 is 0 remainder 3. Next digit is 0: 011000...

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

Adding the last remainder: 01100011

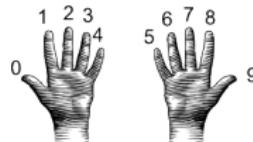
Answer is 1100011.

# Binary to Decimal: Converting Between Bases



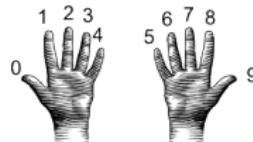
- From binary to decimal:
  - Set sum = last digit.

# Binary to Decimal: Converting Between Bases



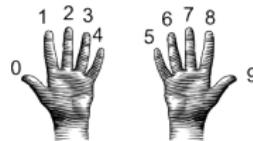
- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by  $2^1$ . Add to sum.

# Binary to Decimal: Converting Between Bases



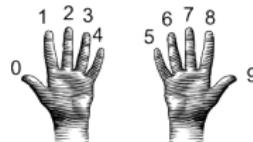
- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by  $2^1$ . Add to sum.
  - Multiply next digit by  $2^2$ . Add to sum.

# Binary to Decimal: Converting Between Bases



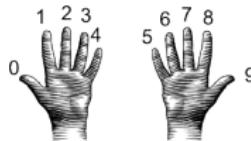
- From binary to decimal:
  - Set sum = last digit.
  - Multiply next digit by  $2 = 2^1$ . Add to sum.
  - Multiply next digit by  $4 = 2^2$ . Add to sum.
  - Multiply next digit by  $8 = 2^3$ . Add to sum.

# Binary to Decimal: Converting Between Bases



- 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.

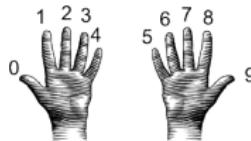
# Binary to Decimal: Converting Between Bases



- From binary to decimal:

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

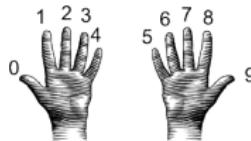
# Binary to Decimal: Converting Between Bases



- From binary to decimal:

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

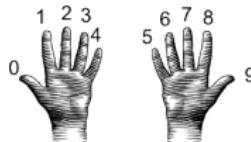
# Binary to Decimal: Converting Between Bases



- 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.

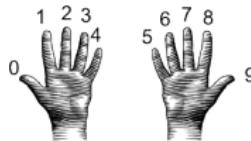
# Binary to Decimal: Converting Between Bases



- 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

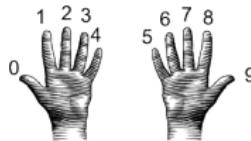


- From binary to decimal:

- 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

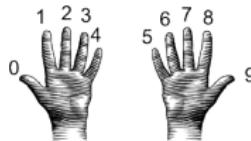


- From binary to decimal:

- 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:

# Binary to Decimal: Converting Between Bases

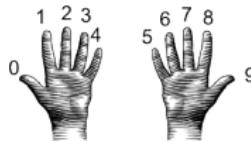


- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



- From binary to decimal:

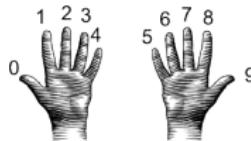
- Set sum = last digit.
- Multiply next digit by  $2^1$ . Add to sum.
- Multiply next digit by  $4 = 2^2$ . Add to sum.
- Multiply next digit by  $8 = 2^3$ . Add to sum.
- Multiply next digit by  $16 = 2^4$ . Add to sum.
- Multiply next digit by  $32 = 2^5$ . Add to sum.
- Multiply next digit by  $64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

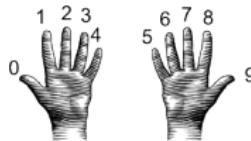


- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



- From binary to decimal:

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

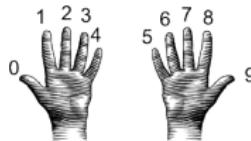
Sum starts with: 1

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

$1 * 4 = 4$ . Add 4 to sum: 5

$1 * 8 = 8$ . Add 8 to sum:

# Binary to Decimal: Converting Between Bases

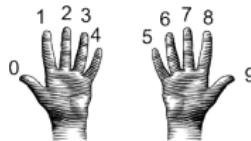


- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases

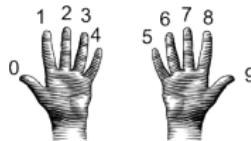


- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

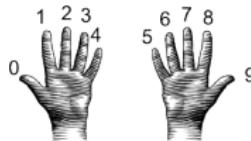


- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

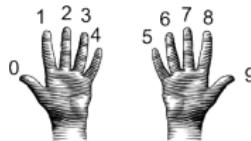


- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases

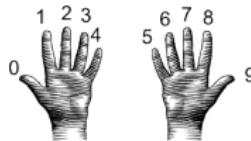


- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

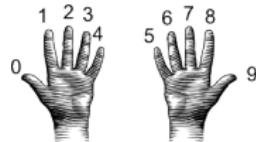


- From binary to decimal:

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

Sum starts with:	1
$0*2 = 0$ . 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

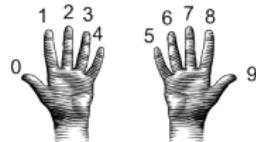
# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with:

# Binary to Decimal: Converting Between Bases

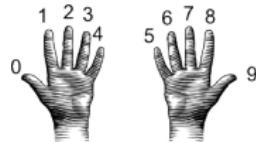


- Example: What is 10100100 in decimal?

Sum starts with: 0

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

# Binary to Decimal: Converting Between Bases

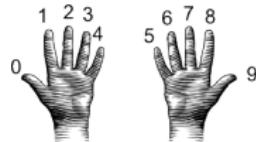


- Example: What is 10100100 in decimal?

Sum starts with: 0

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

# Binary to Decimal: Converting Between Bases



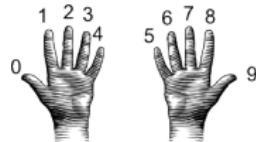
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

# Binary to Decimal: Converting Between Bases



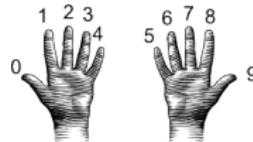
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

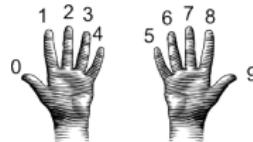
Sum starts with: 0

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

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

$0*8 = 0$ . Add 0 to sum:

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

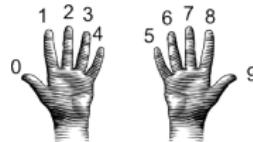
Sum starts with: 0

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

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

$0 * 8 = 0$ . Add 0 to sum: 4

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

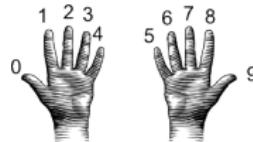
$0 \times 2 = 0$ . Add 0 to sum: 0

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

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

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

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

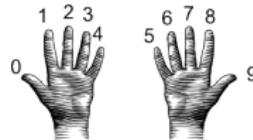
$0 \times 2 = 0$ . Add 0 to sum: 0

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

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

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

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

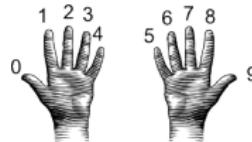
$1 \times 4 = 4$ . Add 4 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

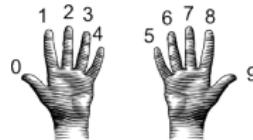
$1 \times 4 = 4$ . Add 4 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

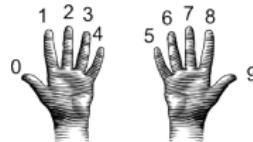
$0 \times 8 = 0$ . Add 0 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

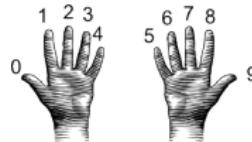
$0 \times 8 = 0$ . Add 0 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

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

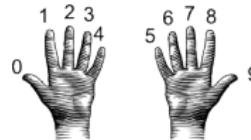
$0 \times 16 = 0$ . Add 0 to sum: 4

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

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

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

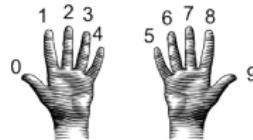
# Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

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

# Binary to Decimal: Converting Between Bases

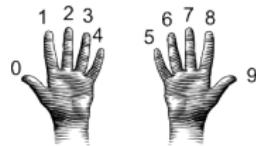


- Example: What is 10100100 in decimal?

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

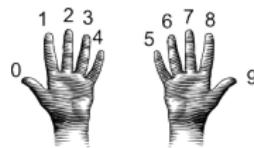
The answer is 164.

# Design Challenge: Incrementers



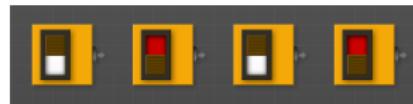
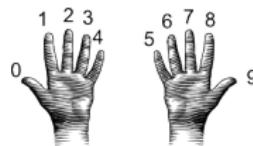
- Simplest arithmetic: add one ("increment") a variable.

# Design Challenge: Incrementers



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

# Design Challenge: Incrementers



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

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

# Design Challenge: Incrementers

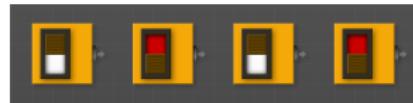
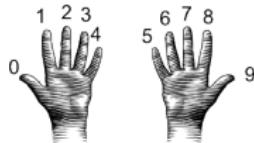


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

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

# Design Challenge: Incrementers

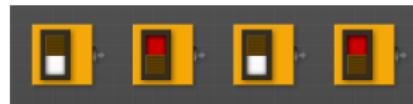
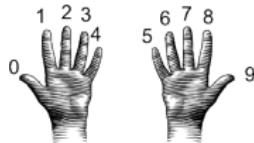


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



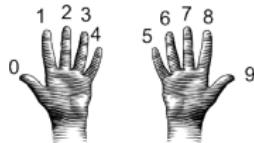
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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

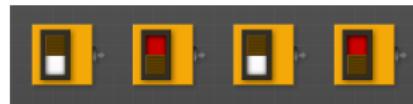
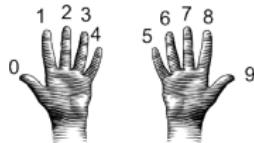


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



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

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

- Challenge: Write an algorithm for incrementing numbers expressed as words.  
Example: "forty one" → "forty two"  
*Hint: Convert to numbers, increment, and convert back to strings.*
- Challenge: Write an algorithm for incrementing binary numbers.  
Example: "1001" → "1010"

# Recap



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

## Recap



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

# Recap



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

# 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 on **the final day of class 9:50a - 12p**

# Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only on **the final day of class 9:50a - 12p**
- The morning of the exam: log into Gradescope, find the **CSci 127 Final Exam** and open the assignment.
- Final Exam will resemble quizzes administered throughout the semester

## 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.
- 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.
- The exam format:
  - ▶ 10 questions, each worth 10 points.

# 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.
- The exam format:
  - ▶ 10 questions, each worth 10 points.
  - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.

# Final Overview: Format

- 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.
- The exam format:
  - ▶ 10 questions, each worth 10 points.
  - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
  - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.

# Final Overview: Format

- 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.
- The exam format:
  - ▶ 10 questions, each worth 10 points.
  - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
  - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
- Mock exam will be available on Gradescope (ungraded, will provide answer keys).