

# CSci 127: Introduction to Computer Science



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

# Frequently Asked Questions

From email and tutoring.

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

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, 23 May, 9-11am.*

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, 23 May, 9-11am.*

*The early final exam (alternative date) is on Friday, 20 May, 8-10am.*

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, 23 May, 9-11am.*

*The early final exam (alternative date) is on Friday, 20 May, 8-10am.*

*Instead of a review sheet, we have:*

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, 23 May, 9-11am.*

*The early final exam (alternative date) is on Friday, 20 May, 8-10am.*

*Instead of a review sheet, we have:*

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

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, 23 May, 9-11am.*

*The early final exam (alternative date) is on Friday, 20 May, 8-10am.*

*Instead of a review sheet, we have:*

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

# Frequently Asked Questions

From email and tutoring.

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

*The official final is Monday, 23 May, 9-11am.*

*The early final exam (alternative date) is on Friday, 20 May, 8-10am.*

*Instead of a review sheet, we have:*

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *UTAs in drop-in tutoring happy to review concepts and old exam questions.*
- ▶ *There will be opportunity for practice during our last meeting on 17 May.*

# Today's Topics



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

# Today's Topics



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

# Predict what the code will do:

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

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

# Python Tutor

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

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

(Demo with pythonTutor)

# Design Pattern: Linear Search

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

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

- Example of **linear search**.

# Design Pattern: Linear Search

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

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

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

# Design Pattern: Linear Search

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

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

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

# Design Pattern: Linear Search

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

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

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

# Today's Topics



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

# Python & Circuits Review: 10 Weeks in 10 Minutes



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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

- As well as definite loops & the turtle package:

The screenshot shows a Python code editor interface. On the left, the code file 'main.py' is open, containing the following Python code:

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

On the right, the 'Result' tab displays the output of the program: a purple hexagon drawn on the screen with black star-shaped stamps at each vertex.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

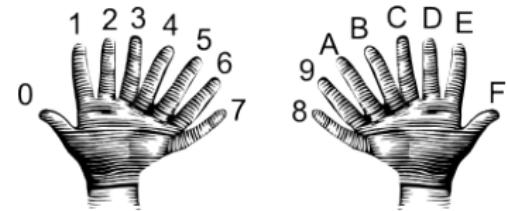
## Week 2: variables, data types, more on loops & range()

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

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

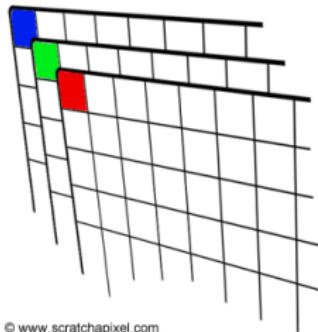
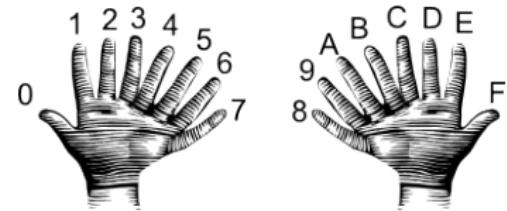
# Week 3: colors, hex, slices, numpy & images

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



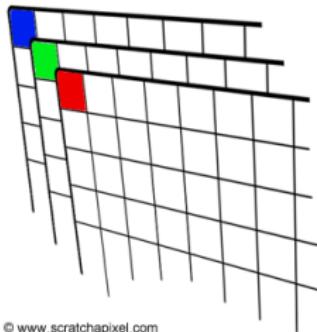
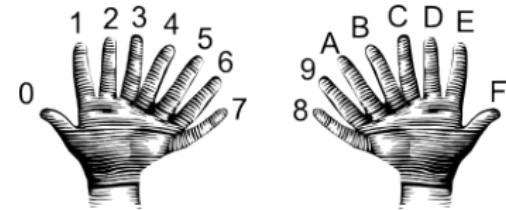
# Week 3: colors, hex, slices, numpy & images

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



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

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



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

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

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

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

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

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



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



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

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



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

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



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

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

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

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

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

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

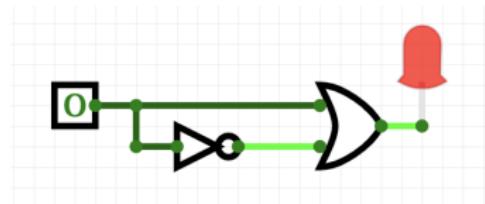
visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

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

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

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

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



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

Source: [https://en.wikipedia.org/wiki/Demographics\\_of\\_New\\_York\\_City](https://en.wikipedia.org/wiki/Demographics_of_New_York_City).....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....

.....  
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,1,037,2037,,727,7881  
1771,21843,36232,,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67541,6250,6840,2035,49734  
1820,123704,11487,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14081,5348,10965,391114  
1850,355441,218013,18951,5815,15821,44115  
1860,513469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1367711,70001,6541,58161,34861,2151134  
1900,1850593,1165852,152999,200567,67621,2437202  
1910,2233142,1634351,284041,430980,8569,4766803  
1920,2211103,2018354,44601,407128,73201,11651,50048  
1930,1867111,1796128,1796128,1796128,1796128,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7891957  
1960,1690101,2319175,1890949,1471277,202055,781984  
1970,1539231,2465701,1874473,1471701,195443,784646  
1980,1426285,2230936,1891325,1168972,252121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,474558,8175133  
2015,1444518,2646733,2339150,1459444,474558,8056405

nycHistPop.csv

In Lab 6

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

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

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  
1771,21843,3623,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,65545,5740,6442,1755,4543,75934  
1820,123704,11487,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,313110,11413,14045,5346,10965,391114  
1850,35549,12891,18951,2895,3546,10965,391115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911801  
1890,1367111,66582,116582,116582,116582,1384534  
1900,185093,116582,152999,200567,67621,2437202  
1910,2233142,1634351,284041,430980,8569,4766803  
1920,22331103,2018354,446071,72201,116582,510488  
1930,16671373,16671373,16671373,16671373,16671373,4930446  
1940,1889924,2469285,1297634,1394711,1374441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,16671373,16671373,16671373,16671373,16671373,781984  
1970,13593231,14657011,14717011,14717011,135443,798460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2004664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2217722,1385108,4175133,8175133  
2015,1444018,2646733,2339150,1459444,474558,8059405

nycHistPop.csv

In Lab 6

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

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Borough,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,2037,...,727,7881  
1771,21843,36231,...,2847,28423  
1790,33131,4549,...,6159,1781,3827,49447  
1800,60515,5740,...,6442,1755,4543,75955  
1810,69031,6200,...,6442,1755,4543,75934  
1820,123704,11187,...,8246,2792,6135,152056  
1830,20589,20535,...,9049,3023,7082,242278  
1840,31510,21013,...,14081,5348,10965,391114  
1850,35541,21881,...,14851,5348,10965,391115  
1860,613469,279122,...,23903,23993,25492,174777  
1870,942292,419921,...,85468,37393,33029,1479103  
1880,1164473,59943,...,56537,51980,33029,1911801  
1890,1367111,70000,...,61861,56537,51980,2341134  
1900,185093,116582,...,152999,200567,67621,2437202  
1910,2331842,1634351,...,28471,430980,8569,476683  
1920,22161103,2018354,...,446071,72021,11651,...,591083  
1930,16671113,...,1579128,...,15821,...,4930446  
1940,1889924,2698285,...,1297634,1394711,1374441,7454995  
1950,1960101,2738175,...,1550949,1451277,191555,...,7991957  
1960,1690101,2738175,...,1689049,1451277,191555,...,7981984  
1970,1539231,...,1687011,...,1472101,...,135443,...,768464  
1980,1426285,...,2230936,...,1891325,...,1168972,...,352121,...,7071639  
1990,1487536,...,2300664,...,1951598,...,1302789,...,1398977,...,7222564  
2000,1537195,...,2485326,...,2229379,...,1332650,...,143782,...,8080879  
2010,1583873,...,2504705,...,2272722,...,1385108,...,143782,...,8175133  
2015,1444018,...,2540733,...,2339150,...,1459444,...,474558,...,8059405
```

nycHistPop.csv

In Lab 6

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

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Population  
1690,203,2037,...,727,7181  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,6350,7000,1800,5300,93734  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31510,21013,14000,5348,10965,391114  
1850,35549,21800,18500,5800,11000,451115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1385000,720000,68000,51800,35000,2100000  
1900,1850093,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44600,720201,11650,500000  
1930,26671128,2203936,1798128,1352454,5832,4930446  
1940,1889924,2690285,1297634,1394711,1374441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690000,2319319,1809000,1400000,1200000,781984  
1970,1539231,2465070,187473,1472701,135443,768460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,379977,7322564  
2000,1537195,2485326,2229379,1332650,419782,8080879  
2010,1583873,2504705,2277722,1385108,4175133,8175133  
2015,1444018,2646733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6

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

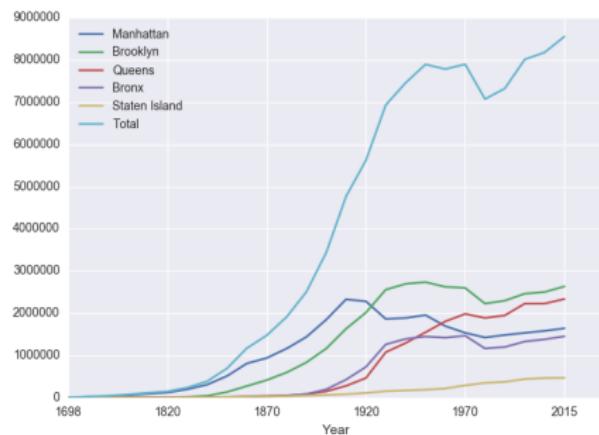
```
import matplotlib.pyplot as plt  
import pandas as pd
```

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Borough,Population  
1698,Manhattan,2037,727,7188  
1771,21843,36231,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,6350,7000,1800,5000,89734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,11013,14000,5348,10965,391114  
1850,355441,128000,18500,58000,11500,450000  
1860,613469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59940,5653,51980,33001,1911801  
1890,1370000,710000,68000,63000,40000,200000  
1900,1850093,116582,152999,200567,67921,2437202  
1910,233142,1634351,2841,430980,8569,476683  
1920,2210103,2018354,44601,73201,11600,500000  
1930,2667103,2487128,35000,35000,35000,500000  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7891957  
1960,1690000,2300000,1800000,1600000,1200000,781984  
1970,1539231,2465000,1471000,1471701,135443,798460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2210722,1385108,451000,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6



# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 7: functions

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

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

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

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

# Week 8: function parameters, github

- Functions can have **input parameters**.

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

# Week 8: function parameters, github

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

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

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

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

# Week 8: function parameters, github

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

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

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

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

# Week 8: function parameters, github

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

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

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

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

# Week 8: function parameters, github

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

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

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

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

# Week 8: function parameters, github

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

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

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

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

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



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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

# Python & Circuits Review: 10 Weeks in 10 Minutes



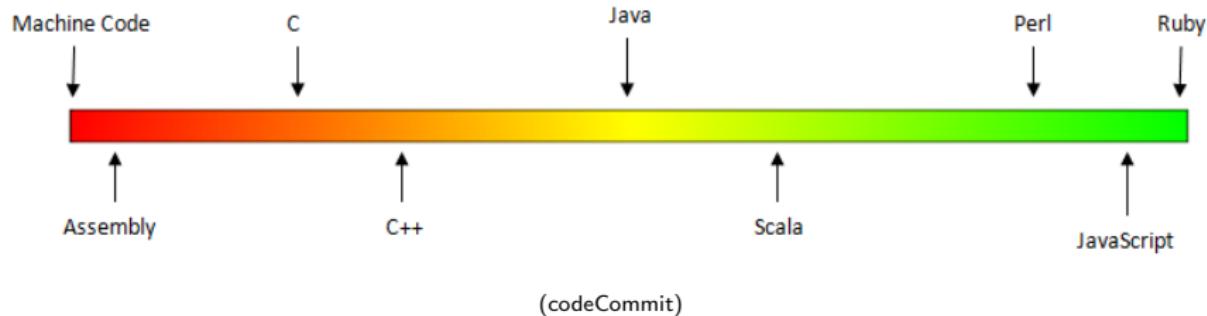
- Input/Output (I/O): `input()` and `print()`; pandas for CSV files
- Types:
  - ▶ Primitive: `int`, `float`, `bool`, `string`;
  - ▶ Container: lists (but not dictionaries/hashes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: if-elif-else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

# Today's Topics



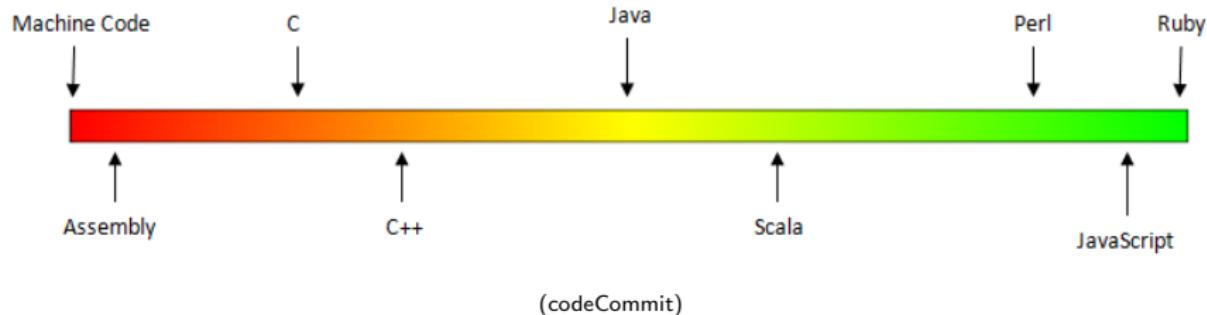
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic

# Low-Level vs. High-Level Languages



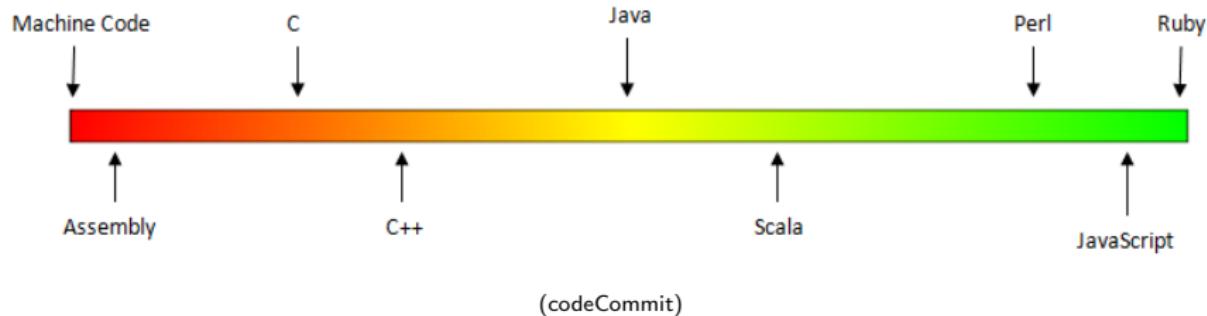
- Can view programming languages on a continuum.

# Low-Level vs. High-Level Languages



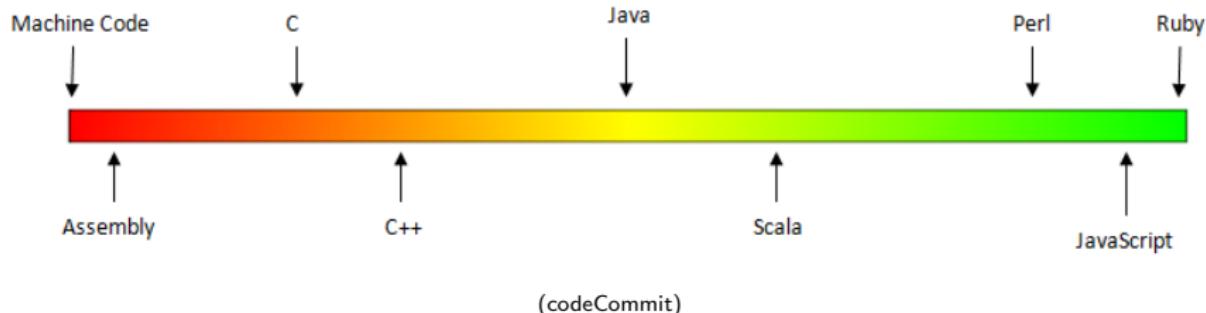
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

# Low-Level vs. High-Level Languages



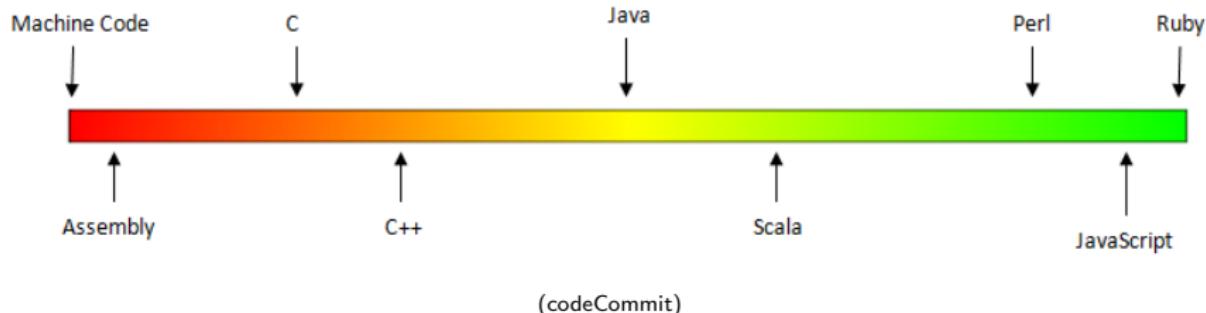
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

# Low-Level vs. High-Level Languages



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

# Low-Level vs. High-Level Languages



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

# Processing

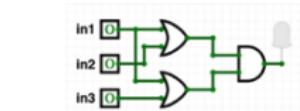
Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.



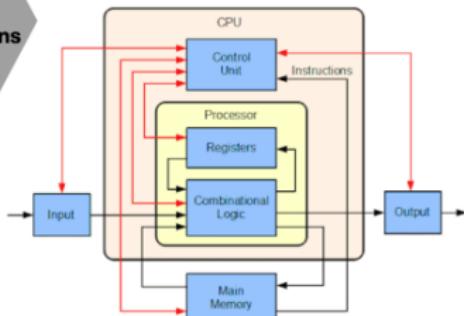
```
111010001110010011000010  
111001000011011011110110111  
001101101000110100010110001  
001011011010000100001111111  
111011011110100011110110111  
0010011010110110100110000001  
001001010101101001000000001  
1111101000000010110011101101  
011000101011011000111010101  
010000100000001010001000000  
011001010101100111001101101  
0011001010101100111001101101  
100101010000001011110011101  
0110111001001000100000001  
1110001000000000001011100  
01100011010111101011011001  
0000011101000110010101110  
011011010110000011010001001  
100100100000000000000000000  
011100110010000000000000000  
100101010000110111010110101  
011101100000000000000000000  
111000110000000000000000000  
011111001001000000000000000  
001000000000000000000000000  
000101010000000000000000000  
011101010000000000000000000  
111011010110011101100100000  
011101010000000000000000000  
000000010110001101100000000
```



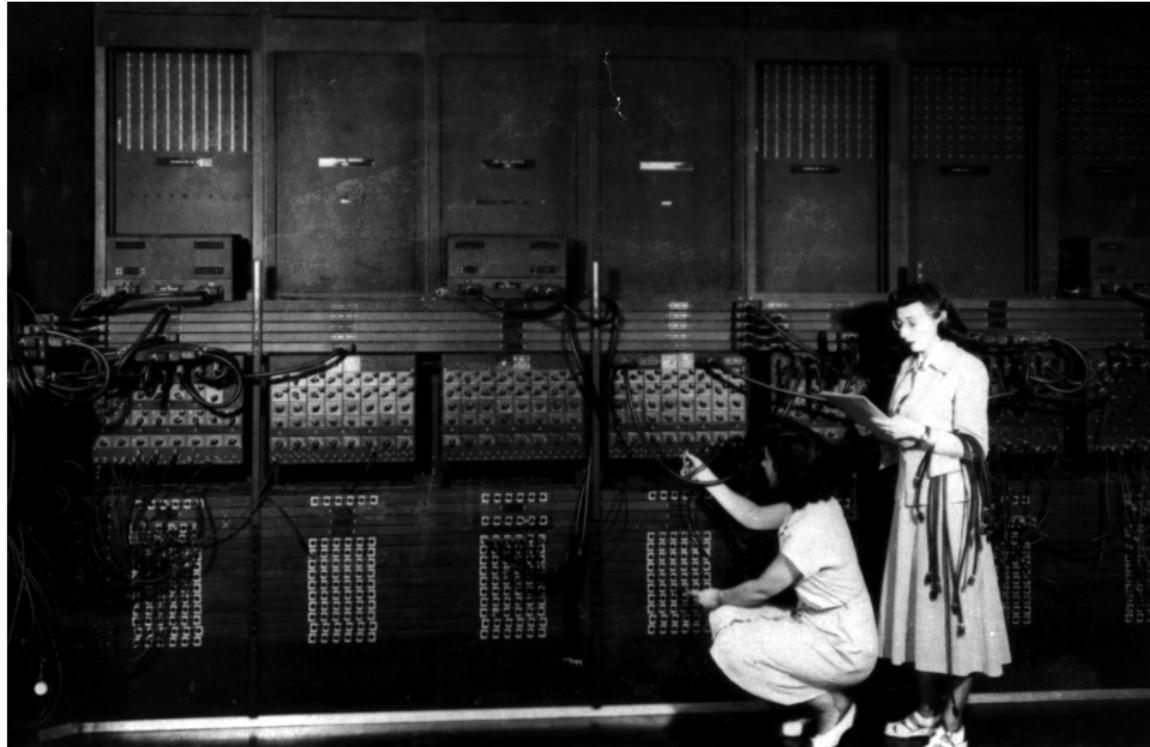
```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)
```



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



# Machine Language



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

# Machine Language

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

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

BREAK

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

(wiki)

# Machine Language

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

(wiki)

# Machine Language

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

(wiki)

# Machine Language

```

A 002300 C2 3B REP #3B
A 002302 7F CLC
A 002303 FB SED
A 002304 34 12 ADD #1224
A 002307 69 21 43 ADC #4321
A 002308 8F B3 77 B1 STA $0017B3
A 00230E D0 CLD
A 00230F E2 3B SEP #3B
A 002311 90 BPK
A 002312

F
PB PC Min:012C A X Y SP DP IR
; 00 2013 00110800 0550 0000 0002 CFFF 0000 00
$ 2000

BREAK

PB PC Min:012C A X Y SP DP IR
; 00 2013 00110800 0550 0000 0002 CFFF 0000 00
$ 77B3 77B3

$0017B3 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

(wiki)

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

# Machine Language



The screenshot shows a terminal window with assembly code and register values. The assembly code is:

```
R 002000 C2 3B      LDR $4, $0
R 002002 1B      SED
R 002003 FB      SCL
R 002004 00        SLO
R 002005 69 34 12    LDR $1234
R 002007 69 21 43    ADC $4321
R 002008 0F 03 7F 01    STA $017F03
R 002009 00        CLD
R 00200F E2 30      SEI
R 002011 00        MRS
R 002012          SWI
```

The registers shown are:

Register	Value
PC	002000
RA	00100000
A	00000000
X	5555 0000
Y	0000 0000
Z	0000 0000
SP	0000 0000
DP	00
SR	0000 0000
CR	0000 0000
LR	0000 0000
PC	002000
RA	00100000
A	00000000
X	5555 0000
Y	0000 0000
Z	0000 0000
SP	0000 0000
DP	00
SR	0000 0000
CR	0000 0000
LR	0000 0000

(wiki)

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

# "Hello World!" in Simplified Machine Language

Line: 3 Go! Show/Hide Demos

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
```

User Guide | Unit Tests | Docs

Step Run  Enable auto switching

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

(WeMIPS)

WeMIPS

## (Demo with WeMIPS)

# MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there's a menu bar with 'File', 'Edit', 'Run', 'Help', 'Show/Hide Demo', 'Addition', 'Subtraction', 'Multiplication', 'Division', 'Hello World', 'Code Gen Base String', 'Interactive', 'Binary Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with 'Step', 'Run', 'Break', 'Create auto watching', and 'Stop' buttons. A status bar at the bottom right shows navigation icons.

The main area contains assembly code:

```
1 # Shows "Hello world!" at the top of the stack
2
3 .text
4 .globl _start
5 _start:
6    li $t0, 8192
7    li $t1, 111901
8    li $t2, 111901
9    li $t3, 111901
10   li $t4, 111901
11   li $t5, 111901
12   li $t6, 111901
13   li $t7, 111901
14   li $t8, 111901
15   li $t9, 111901
16   li $t10, 111901
17   li $t11, 111901
18   li $t12, 111901
19   li $t13, 111901
20   li $t14, 111901
21   li $t15, 111901
22   li $t16, 111901
23   li $t17, 111901
24   li $t18, 1024000
25   li $t19, 1024000
26   li $t20, 111901
27   li $t21, 111901
28   li $t22, 111901
29   li $t23, 111901
30   li $t24, 111901
31   li $t25, 111901
32   li $t26, 111901
33   li $t27, 111901
34   li $t28, 111901
35   li $t29, 111901
36   li $t30, 111901
37   li $t31, 111901
38
39   add $t0, $t1, $t2
40   add $t0, $t3, $t4
41   add $t0, $t5, $t6
42   add $t0, $t7, $t8
43   add $t0, $t9, $t10
44   add $t0, $t11, $t12
45   add $t0, $t13, $t14
46   add $t0, $t15, $t16
47   add $t0, $t17, $t18
48   add $t0, $t19, $t20
49   add $t0, $t21, $t22
50   add $t0, $t23, $t24
51   add $t0, $t25, $t26
52   add $t0, $t27, $t28
53   add $t0, $t29, $t30
54   add $t0, $t31, $t30
55
56   add $t0, $t1, $t2
57   add $t0, $t3, $t4
58   add $t0, $t5, $t6
59   add $t0, $t7, $t8
60   add $t0, $t9, $t10
61   add $t0, $t11, $t12
62   add $t0, $t13, $t14
63   add $t0, $t15, $t16
64   add $t0, $t17, $t18
65   add $t0, $t19, $t20
66   add $t0, $t21, $t22
67   add $t0, $t23, $t24
68   add $t0, $t25, $t26
69   add $t0, $t27, $t28
70   add $t0, $t29, $t30
71   add $t0, $t31, $t30
72
73   add $t0, $t1, $t2
74   add $t0, $t3, $t4
75   add $t0, $t5, $t6
76   add $t0, $t7, $t8
77   add $t0, $t9, $t10
78   add $t0, $t11, $t12
79   add $t0, $t13, $t14
80   add $t0, $t15, $t16
81   add $t0, $t17, $t18
82   add $t0, $t19, $t20
83   add $t0, $t21, $t22
84   add $t0, $t23, $t24
85   add $t0, $t25, $t26
86   add $t0, $t27, $t28
87   add $t0, $t29, $t30
88   add $t0, $t31, $t30
89
90   add $t0, $t1, $t2
91   add $t0, $t3, $t4
92   add $t0, $t5, $t6
93   add $t0, $t7, $t8
94   add $t0, $t9, $t10
95   add $t0, $t11, $t12
96   add $t0, $t13, $t14
97   add $t0, $t15, $t16
98   add $t0, $t17, $t18
99   add $t0, $t19, $t20
100  add $t0, $t21, $t22
101  add $t0, $t23, $t24
102  add $t0, $t25, $t26
103  add $t0, $t27, $t28
104  add $t0, $t29, $t30
105  add $t0, $t31, $t30
106
107  add $t0, $t1, $t2
108  add $t0, $t3, $t4
109  add $t0, $t5, $t6
110  add $t0, $t7, $t8
111  add $t0, $t9, $t10
112  add $t0, $t11, $t12
113  add $t0, $t13, $t14
114  add $t0, $t15, $t16
115  add $t0, $t17, $t18
116  add $t0, $t19, $t20
117  add $t0, $t21, $t22
118  add $t0, $t23, $t24
119  add $t0, $t25, $t26
120  add $t0, $t27, $t28
121  add $t0, $t29, $t30
122  add $t0, $t31, $t30
123
124  add $t0, $t1, $t2
125  add $t0, $t3, $t4
126  add $t0, $t5, $t6
127  add $t0, $t7, $t8
128  add $t0, $t9, $t10
129  add $t0, $t11, $t12
130  add $t0, $t13, $t14
131  add $t0, $t15, $t16
132  add $t0, $t17, $t18
133  add $t0, $t19, $t20
134  add $t0, $t21, $t22
135  add $t0, $t23, $t24
136  add $t0, $t25, $t26
137  add $t0, $t27, $t28
138  add $t0, $t29, $t30
139  add $t0, $t31, $t30
140
141  add $t0, $t1, $t2
142  add $t0, $t3, $t4
143  add $t0, $t5, $t6
144  add $t0, $t7, $t8
145  add $t0, $t9, $t10
146  add $t0, $t11, $t12
147  add $t0, $t13, $t14
148  add $t0, $t15, $t16
149  add $t0, $t17, $t18
150  add $t0, $t19, $t20
151  add $t0, $t21, $t22
152  add $t0, $t23, $t24
153  add $t0, $t25, $t26
154  add $t0, $t27, $t28
155  add $t0, $t29, $t30
156  add $t0, $t31, $t30
157
158  add $t0, $t1, $t2
159  add $t0, $t3, $t4
160  add $t0, $t5, $t6
161  add $t0, $t7, $t8
162  add $t0, $t9, $t10
163  add $t0, $t11, $t12
164  add $t0, $t13, $t14
165  add $t0, $t15, $t16
166  add $t0, $t17, $t18
167  add $t0, $t19, $t20
168  add $t0, $t21, $t22
169  add $t0, $t23, $t24
170  add $t0, $t25, $t26
171  add $t0, $t27, $t28
172  add $t0, $t29, $t30
173  add $t0, $t31, $t30
174
175  add $t0, $t1, $t2
176  add $t0, $t3, $t4
177  add $t0, $t5, $t6
178  add $t0, $t7, $t8
179  add $t0, $t9, $t10
180  add $t0, $t11, $t12
181  add $t0, $t13, $t14
182  add $t0, $t15, $t16
183  add $t0, $t17, $t18
184  add $t0, $t19, $t20
185  add $t0, $t21, $t22
186  add $t0, $t23, $t24
187  add $t0, $t25, $t26
188  add $t0, $t27, $t28
189  add $t0, $t29, $t30
190  add $t0, $t31, $t30
191
192  add $t0, $t1, $t2
193  add $t0, $t3, $t4
194  add $t0, $t5, $t6
195  add $t0, $t7, $t8
196  add $t0, $t9, $t10
197  add $t0, $t11, $t12
198  add $t0, $t13, $t14
199  add $t0, $t15, $t16
200  add $t0, $t17, $t18
201  add $t0, $t19, $t20
202  add $t0, $t21, $t22
203  add $t0, $t23, $t24
204  add $t0, $t25, $t26
205  add $t0, $t27, $t28
206  add $t0, $t29, $t30
207  add $t0, $t31, $t30
208
209  add $t0, $t1, $t2
210  add $t0, $t3, $t4
211  add $t0, $t5, $t6
212  add $t0, $t7, $t8
213  add $t0, $t9, $t10
214  add $t0, $t11, $t12
215  add $t0, $t13, $t14
216  add $t0, $t15, $t16
217  add $t0, $t17, $t18
218  add $t0, $t19, $t20
219  add $t0, $t21, $t22
220  add $t0, $t23, $t24
221  add $t0, $t25, $t26
222  add $t0, $t27, $t28
223  add $t0, $t29, $t30
224  add $t0, $t31, $t30
225
226  add $t0, $t1, $t2
227  add $t0, $t3, $t4
228  add $t0, $t5, $t6
229  add $t0, $t7, $t8
230  add $t0, $t9, $t10
231  add $t0, $t11, $t12
232  add $t0, $t13, $t14
233  add $t0, $t15, $t16
234  add $t0, $t17, $t18
235  add $t0, $t19, $t20
236  add $t0, $t21, $t22
237  add $t0, $t23, $t24
238  add $t0, $t25, $t26
239  add $t0, $t27, $t28
240  add $t0, $t29, $t30
241  add $t0, $t31, $t30
242
243  add $t0, $t1, $t2
244  add $t0, $t3, $t4
245  add $t0, $t5, $t6
246  add $t0, $t7, $t8
247  add $t0, $t9, $t10
248  add $t0, $t11, $t12
249  add $t0, $t13, $t14
250  add $t0, $t15, $t16
251  add $t0, $t17, $t18
252  add $t0, $t19, $t20
253  add $t0, $t21, $t22
254  add $t0, $t23, $t24
255  add $t0, $t25, $t26
256  add $t0, $t27, $t28
257  add $t0, $t29, $t30
258  add $t0, $t31, $t30
259
260  add $t0, $t1, $t2
261  add $t0, $t3, $t4
262  add $t0, $t5, $t6
263  add $t0, $t7, $t8
264  add $t0, $t9, $t10
265  add $t0, $t11, $t12
266  add $t0, $t13, $t14
267  add $t0, $t15, $t16
268  add $t0, $t17, $t18
269  add $t0, $t19, $t20
270  add $t0, $t21, $t22
271  add $t0, $t23, $t24
272  add $t0, $t25, $t26
273  add $t0, $t27, $t28
274  add $t0, $t29, $t30
275  add $t0, $t31, $t30
276
277  add $t0, $t1, $t2
278  add $t0, $t3, $t4
279  add $t0, $t5, $t6
280  add $t0, $t7, $t8
281  add $t0, $t9, $t10
282  add $t0, $t11, $t12
283  add $t0, $t13, $t14
284  add $t0, $t15, $t16
285  add $t0, $t17, $t18
286  add $t0, $t19, $t20
287  add $t0, $t21, $t22
288  add $t0, $t23, $t24
289  add $t0, $t25, $t26
290  add $t0, $t27, $t28
291  add $t0, $t29, $t30
292  add $t0, $t31, $t30
293
294  add $t0, $t1, $t2
295  add $t0, $t3, $t4
296  add $t0, $t5, $t6
297  add $t0, $t7, $t8
298  add $t0, $t9, $t10
299  add $t0, $t11, $t12
300  add $t0, $t13, $t14
301  add $t0, $t15, $t16
302  add $t0, $t17, $t18
303  add $t0, $t19, $t20
304  add $t0, $t21, $t22
305  add $t0, $t23, $t24
306  add $t0, $t25, $t26
307  add $t0, $t27, $t28
308  add $t0, $t29, $t30
309  add $t0, $t31, $t30
310
311  add $t0, $t1, $t2
312  add $t0, $t3, $t4
313  add $t0, $t5, $t6
314  add $t0, $t7, $t8
315  add $t0, $t9, $t10
316  add $t0, $t11, $t12
317  add $t0, $t13, $t14
318  add $t0, $t15, $t16
319  add $t0, $t17, $t18
320  add $t0, $t19, $t20
321  add $t0, $t21, $t22
322  add $t0, $t23, $t24
323  add $t0, $t25, $t26
324  add $t0, $t27, $t28
325  add $t0, $t29, $t30
326  add $t0, $t31, $t30
327
328  add $t0, $t1, $t2
329  add $t0, $t3, $t4
330  add $t0, $t5, $t6
331  add $t0, $t7, $t8
332  add $t0, $t9, $t10
333  add $t0, $t11, $t12
334  add $t0, $t13, $t14
335  add $t0, $t15, $t16
336  add $t0, $t17, $t18
337  add $t0, $t19, $t20
338  add $t0, $t21, $t22
339  add $t0, $t23, $t24
340  add $t0, $t25, $t26
341  add $t0, $t27, $t28
342  add $t0, $t29, $t30
343  add $t0, $t31, $t30
344
345  add $t0, $t1, $t2
346  add $t0, $t3, $t4
347  add $t0, $t5, $t6
348  add $t0, $t7, $t8
349  add $t0, $t9, $t10
350  add $t0, $t11, $t12
351  add $t0, $t13, $t14
352  add $t0, $t15, $t16
353  add $t0, $t17, $t18
354  add $t0, $t19, $t20
355  add $t0, $t21, $t22
356  add $t0, $t23, $t24
357  add $t0, $t25, $t26
358  add $t0, $t27, $t28
359  add $t0, $t29, $t30
360  add $t0, $t31, $t30
361
362  add $t0, $t1, $t2
363  add $t0, $t3, $t4
364  add $t0, $t5, $t6
365  add $t0, $t7, $t8
366  add $t0, $t9, $t10
367  add $t0, $t11, $t12
368  add $t0, $t13, $t14
369  add $t0, $t15, $t16
370  add $t0, $t17, $t18
371  add $t0, $t19, $t20
372  add $t0, $t21, $t22
373  add $t0, $t23, $t24
374  add $t0, $t25, $t26
375  add $t0, $t27, $t28
376  add $t0, $t29, $t30
377  add $t0, $t31, $t30
378
379  add $t0, $t1, $t2
380  add $t0, $t3, $t4
381  add $t0, $t5, $t6
382  add $t0, $t7, $t8
383  add $t0, $t9, $t10
384  add $t0, $t11, $t12
385  add $t0, $t13, $t14
386  add $t0, $t15, $t16
387  add $t0, $t17, $t18
388  add $t0, $t19, $t20
389  add $t0, $t21, $t22
390  add $t0, $t23, $t24
391  add $t0, $t25, $t26
392  add $t0, $t27, $t28
393  add $t0, $t29, $t30
394  add $t0, $t31, $t30
395
396  add $t0, $t1, $t2
397  add $t0, $t3, $t4
398  add $t0, $t5, $t6
399  add $t0, $t7, $t8
400  add $t0, $t9, $t10
401  add $t0, $t11, $t12
402  add $t0, $t13, $t14
403  add $t0, $t15, $t16
404  add $t0, $t17, $t18
405  add $t0, $t19, $t20
406  add $t0, $t21, $t22
407  add $t0, $t23, $t24
408  add $t0, $t25, $t26
409  add $t0, $t27, $t28
410  add $t0, $t29, $t30
411  add $t0, $t31, $t30
412
413  add $t0, $t1, $t2
414  add $t0, $t3, $t4
415  add $t0, $t5, $t6
416  add $t0, $t7, $t8
417  add $t0, $t9, $t10
418  add $t0, $t11, $t12
419  add $t0, $t13, $t14
420  add $t0, $t15, $t16
421  add $t0, $t17, $t18
422  add $t0, $t19, $t20
423  add $t0, $t21, $t22
424  add $t0, $t23, $t24
425  add $t0, $t25, $t26
426  add $t0, $t27, $t28
427  add $t0, $t29, $t30
428  add $t0, $t31, $t30
429
430  add $t0, $t1, $t2
431  add $t0, $t3, $t4
432  add $t0, $t5, $t6
433  add $t0, $t7, $t8
434  add $t0, $t9, $t10
435  add $t0, $t11, $t12
436  add $t0, $t13, $t14
437  add $t0, $t15, $t16
438  add $t0, $t17, $t18
439  add $t0, $t19, $t20
440  add $t0, $t21, $t22
441  add $t0, $t23, $t24
442  add $t0, $t25, $t26
443  add $t0, $t27, $t28
444  add $t0, $t29, $t30
445  add $t0, $t31, $t30
446
447  add $t0, $t1, $t2
448  add $t0, $t3, $t4
449  add $t0, $t5, $t6
450  add $t0, $t7, $t8
451  add $t0, $t9, $t10
452  add $t0, $t11, $t12
453  add $t0, $t13, $t14
454  add $t0, $t15, $t16
455  add $t0, $t17, $t18
456  add $t0, $t19, $t20
457  add $t0, $t21, $t22
458  add $t0, $t23, $t24
459  add $t0, $t25, $t26
460  add $t0, $t27, $t28
461  add $t0, $t29, $t30
462  add $t0, $t31, $t30
463
464  add $t0, $t1, $t2
465  add $t0, $t3, $t4
466  add $t0, $t5, $t6
467  add $t0, $t7, $t8
468  add $t0, $t9, $t10
469  add $t0, $t11, $t12
470  add $t0, $t13, $t14
471  add $t0, $t15, $t16
472  add $t0, $t17, $t18
473  add $t0, $t19, $t20
474  add $t0, $t21, $t22
475  add $t0, $t23, $t24
476  add $t0, $t25, $t26
477  add $t0, $t27, $t28
478  add $t0, $t29, $t30
479  add $t0, $t31, $t30
480
481  add $t0, $t1, $t2
482  add $t0, $t3, $t4
483  add $t0, $t5, $t6
484  add $t0, $t7, $t8
485  add $t0, $t9, $t10
486  add $t0, $t11, $t12
487  add $t0, $t13, $t14
488  add $t0, $t15, $t16
489  add $t0, $t17, $t18
490  add $t0, $t19, $t20
491  add $t0, $t21, $t22
492  add $t0, $t23, $t24
493  add $t0, $t25, $t26
494  add $t0, $t27, $t28
495  add $t0, $t29, $t30
496  add $t0, $t31, $t30
497
498  add $t0, $t1, $t2
499  add $t0, $t3, $t4
500  add $t0, $t5, $t6
501  add $t0, $t7, $t8
502  add $t0, $t9, $t10
503  add $t0, $t11, $t12
504  add $t0, $t13, $t14
505  add $t0, $t15, $t16
506  add $t0, $t17, $t18
507  add $t0, $t19, $t20
508  add $t0, $t21, $t22
509  add $t0, $t23, $t24
510  add $t0, $t25, $t26
511  add $t0, $t27, $t28
512  add $t0, $t29, $t30
513  add $t0, $t31, $t30
514
515  add $t0, $t1, $t2
516  add $t0, $t3, $t4
517  add $t0, $t5, $t6
518  add $t0, $t7, $t8
519  add $t0, $t9, $t10
520  add $t0, $t11, $t12
521  add $t0, $t13, $t14
522  add $t0, $t15, $t16
523  add $t0, $t17, $t18
524  add $t0, $t19, $t20
525  add $t0, $t21, $t22
526  add $t0, $t23, $t24
527  add $t0, $t25, $t26
528  add $t0, $t27, $t28
529  add $t0, $t29, $t30
530  add $t0, $t31, $t30
531
532  add $t0, $t1, $t2
533  add $t0, $t3, $t4
534  add $t0, $t5, $t6
535  add $t0, $t7, $t8
536  add $t0, $t9, $t10
537  add $t0, $t11, $t12
538  add $t0, $t13, $t14
539  add $t0, $t15, $t16
540  add $t0, $t17, $t18
541  add $t0, $t19, $t20
542  add $t0, $t21, $t22
543  add $t0, $t23, $t24
544  add $t0, $t25, $t26
545  add $t0, $t27, $t28
546  add $t0, $t29, $t30
547  add $t0, $t31, $t30
548
549  add $t0, $t1, $t2
550  add $t0, $t3, $t4
551  add $t0, $t5, $t6
552  add $t0, $t7, $t8
553  add $t0, $t9, $t10
554  add $t0, $t11, $t12
555  add $t0, $t13, $t14
556  add $t0, $t15, $t16
557  add $t0, $t17, $t18
558  add $t0, $t19, $t20
559  add $t0, $t21, $t22
560  add $t0, $t23, $t24
561  add $t0, $t25, $t26
562  add $t0, $t27, $t28
563  add $t0, $t29, $t30
564  add $t0, $t31, $t30
565
566  add $t0, $t1, $t2
567  add $t0, $t3, $t4
568  add $t0, $t5, $t6
569  add $t0, $t7, $t8
570  add $t0, $t9, $t10
571  add $t0, $t11, $t12
572  add $t0, $t13, $t14
573  add $t0, $t15, $t16
574  add $t0, $t17, $t18
575  add $t0, $t19, $t20
576  add $t0, $t21, $t22
577  add $t0, $t23, $t24
578  add $t0, $t25, $t26
579  add $t0, $t27, $t28
580  add $t0, $t29, $t30
581  add $t0, $t31, $t30
582
583  add $t0, $t1, $t2
584  add $t0, $t3, $t4
585  add $t0, $t5, $t6
586  add $t0, $t7, $t8
587  add $t0, $t9, $t10
588  add $t0, $t11, $t12
589  add $t0, $t13, $t14
590  add $t0, $t15, $t16
591  add $t0, $t17, $t18
592  add $t0, $t19, $t20
593  add $t0, $t21, $t22
594  add $t0, $t23, $t24
595  add $t0, $t25, $t26
596  add $t0, $t27, $t28
597  add $t0, $t29, $t30
598  add $t0, $t31, $t30
599
510  add $t0, $t1, $t2
511  add $t0, $t3, $t4
512  add $t0, $t5, $t6
513  add $t0, $t7, $t8
514  add $t0, $t9, $t10
515  add $t0, $t11, $t12
516  add $t0, $t13, $t14
517  add $t0, $t15, $t16
518  add $t0, $t17, $t18
519  add $t0, $t19, $t20
520  add $t0, $t21, $t22
521  add $t0, $t23, $t24
522  add $t0, $t25, $t26
523  add $t0, $t27, $t28
524  add $t0, $t29, $t30
525  add $t0, $t31, $t30
526
527  add $t0, $t1, $t2
528  add $t0, $t3, $t4
529  add $t0, $t5, $t6
530  add $t0, $t7, $t8
531  add $t0, $t9, $t10
532  add $t0, $t11, $t12
533  add $t0, $t13, $t14
534  add $t0, $t15, $t16
535  add $t0, $t17, $t18
536  add $t0, $t19, $t20
537  add $t0, $t21, $t22
538  add $t0, $t23, $t24
539  add $t0, $t25, $t26
540  add $t0, $t27, $t28
541  add $t0, $t29, $t30
542  add $t0, $t31, $t30
543
544  add $t0, $t1, $t2
545  add $t0, $t3, $t4
546  add $t0, $t5, $t6
547  add $t0, $t7, $t8
548  add $t0, $t9, $t10
549  add $t0, $t11, $t12
550  add $t0, $t13, $t14
551  add $t0, $t15, $t16
552  add $t0, $t17, $t18
553  add $t0, $t19, $t20
554  add $t0, $t21, $t22
555  add $t0, $t23, $t24
556  add $t0, $t25, $t26
557  add $t0, $t27, $t28
558  add $t0, $t29, $t30
559  add $t0, $t31, $t30
560
561  add $t0, $t1, $t2
562  add $t0, $t3, $t4
563  add $t0, $t5, $t6
564  add $t0, $t7, $t8
565  add $t0, $t9, $t10
566  add $t0, $t11, $t12
567  add $t0, $t13, $t14
568  add $t0, $t15, $t16
569  add $t0, $t17, $t18
570  add $t0, $t19, $t20
571  add $t0, $t21, $t22
572  add $t0, $t23, $t24
573  add $t0, $t25, $t26
574  add $t0, $t27, $t28
575  add $t0, $t29, $t30
576  add $t0, $t31, $t30
577
578  add $t0, $t1, $t2
579  add $t0, $t3, $t4
580  add $t0, $t5, $t6
581  add $t0, $t7, $t8
582  add $t0, $t9, $t10
583  add $t0, $t11, $t12
584  add $t0, $t13, $t14
585  add $t0, $t15, $t16
586  add $t0, $t17, $t18
587  add $t0, $t19, $t20
588  add $t0, $t21, $t22
589  add $t0, $t23, $t24
590  add $t0, $t25, $t26
591  add $t0, $t27, $t28
592  add $t0, $t29, $t30
593  add $t0, $t31, $t30
594
595  add $t0, $t1, $t2
596  add $t0, $t3, $t4
597  add $t0, $t5, $t6
598  add $t0, $t7, $t8
599  add $t0, $t9, $t10
600  add $t0, $t11, $t12
601  add $t0, $t13, $t14
602  add $t0, $t15, $t16
603  add $t0, $t17, $t18
604  add $t0, $t19, $t20
605  add $t0, $t21, $t22
606  add $t0, $t23, $t24
607  add $t0, $t25, $t26
608  add $t0, $t27, $t28
609  add $t0, $t29, $t30
610  add $t0, $t31, $t30
611
612  add $t0,
```

# MIPS Commands

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

## MIPS Commands

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

## MIPS Commands

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

## MIPS Commands

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

## MIPS Commands

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

# MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there's a menu bar with 'File', 'Edit', 'Get', 'Show/Hide Demo', 'User Guide | Unit Tests | Docs', and tabs for 'Assembly', 'Debugger', 'Hex', 'Looper', 'Stack View', 'Hello World', 'Code Gen', 'Save String', 'Interactive', 'Decimal Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with 'Step', 'Run', 'Break', 'Create auto watching', and 'Stop' buttons. A status bar at the bottom shows memory addresses from \$0 to \$100.

The assembly code area contains the following instructions:

```
1 # Shows 'Hello world' at the top of the stack
2 .text
3 .globl _start
4 .data
5 _start: .asciz "Hello world\n"
6 .text
7 _start: li $t0, _start
8 sb $t0, $11($sp)
9 li $t1, 11
10 sb $t1, $11($sp)
11 li $t2, 0
12 sb $t2, $11($sp)
13 li $t3, 11
14 sb $t3, $11($sp)
15 li $t4, 11
16 sb $t4, $11($sp)
17 li $t5, 11
18 sb $t5, $11($sp)
19 li $t6, 11
20 sb $t6, $11($sp)
21 li $t7, 11
22 sb $t7, $11($sp)
23 li $t8, 11
24 sb $t8, $11($sp)
25 li $t9, 11
26 sb $t9, $11($sp)
27 li $t10, 11
28 sb $t10, $11($sp)
29 li $t11, 11
30 sb $t11, $11($sp)
31 addi $t12, $t11, 111 # $t12 = $t11 + 111
32 addi $t13, $t12, 111 # $t13 = $t12 + 111
33 addi $t14, $t13, 111 # $t14 = $t13 + 111
34 addi $t15, $t14, 111 # $t15 = $t14 + 111
35 addi $t16, $t15, 111 # $t16 = $t15 + 111
36 addi $t17, $t16, 111 # $t17 = $t16 + 111
37 addi $t18, $t17, 111 # $t18 = $t17 + 111
38 addi $t19, $t18, 111 # $t19 = $t18 + 111
39 addi $t20, $t19, 111 # $t20 = $t19 + 111
40 addi $t21, $t20, 111 # $t21 = $t20 + 111
41 addi $t22, $t21, 111 # $t22 = $t21 + 111
42 addi $t23, $t22, 111 # $t23 = $t22 + 111
43 addi $t24, $t23, 111 # $t24 = $t23 + 111
44 addi $t25, $t24, 111 # $t25 = $t24 + 111
45 addi $t26, $t25, 111 # $t26 = $t25 + 111
46 addi $t27, $t26, 111 # $t27 = $t26 + 111
47 addi $t28, $t27, 111 # $t28 = $t27 + 111
48 addi $t29, $t28, 111 # $t29 = $t28 + 111
49 addi $t30, $t29, 111 # $t30 = $t29 + 111
50 addi $t31, $t30, 111 # $t31 = $t30 + 111
51 addi $t32, $t31, 111 # $t32 = $t31 + 111
52 addi $t33, $t32, 111 # $t33 = $t32 + 111
53 addi $t34, $t33, 111 # $t34 = $t33 + 111
54 addi $t35, $t34, 111 # $t35 = $t34 + 111
55 addi $t36, $t35, 111 # $t36 = $t35 + 111
56 addi $t37, $t36, 111 # $t37 = $t36 + 111
57 addi $t38, $t37, 111 # $t38 = $t37 + 111
58 addi $t39, $t38, 111 # $t39 = $t38 + 111
59 addi $t40, $t39, 111 # $t40 = $t39 + 111
60 addi $t41, $t40, 111 # $t41 = $t40 + 111
61 addi $t42, $t41, 111 # $t42 = $t41 + 111
62 addi $t43, $t42, 111 # $t43 = $t42 + 111
63 addi $t44, $t43, 111 # $t44 = $t43 + 111
64 addi $t45, $t44, 111 # $t45 = $t44 + 111
65 addi $t46, $t45, 111 # $t46 = $t45 + 111
66 addi $t47, $t46, 111 # $t47 = $t46 + 111
67 addi $t48, $t47, 111 # $t48 = $t47 + 111
68 addi $t49, $t48, 111 # $t49 = $t48 + 111
69 addi $t50, $t49, 111 # $t50 = $t49 + 111
70 addi $t51, $t50, 111 # $t51 = $t50 + 111
71 addi $t52, $t51, 111 # $t52 = $t51 + 111
72 addi $t53, $t52, 111 # $t53 = $t52 + 111
73 addi $t54, $t53, 111 # $t54 = $t53 + 111
74 addi $t55, $t54, 111 # $t55 = $t54 + 111
75 addi $t56, $t55, 111 # $t56 = $t55 + 111
76 addi $t57, $t56, 111 # $t57 = $t56 + 111
77 addi $t58, $t57, 111 # $t58 = $t57 + 111
78 addi $t59, $t58, 111 # $t59 = $t58 + 111
79 addi $t60, $t59, 111 # $t60 = $t59 + 111
80 addi $t61, $t60, 111 # $t61 = $t60 + 111
81 addi $t62, $t61, 111 # $t62 = $t61 + 111
82 addi $t63, $t62, 111 # $t63 = $t62 + 111
83 addi $t64, $t63, 111 # $t64 = $t63 + 111
84 addi $t65, $t64, 111 # $t65 = $t64 + 111
85 addi $t66, $t65, 111 # $t66 = $t65 + 111
86 addi $t67, $t66, 111 # $t67 = $t66 + 111
87 addi $t68, $t67, 111 # $t68 = $t67 + 111
88 addi $t69, $t68, 111 # $t69 = $t68 + 111
89 addi $t70, $t69, 111 # $t70 = $t69 + 111
90 addi $t71, $t70, 111 # $t71 = $t70 + 111
91 addi $t72, $t71, 111 # $t72 = $t71 + 111
92 addi $t73, $t72, 111 # $t73 = $t72 + 111
93 addi $t74, $t73, 111 # $t74 = $t73 + 111
94 addi $t75, $t74, 111 # $t75 = $t74 + 111
95 addi $t76, $t75, 111 # $t76 = $t75 + 111
96 addi $t77, $t76, 111 # $t77 = $t76 + 111
97 addi $t78, $t77, 111 # $t78 = $t77 + 111
98 addi $t79, $t78, 111 # $t79 = $t78 + 111
99 addi $t80, $t79, 111 # $t80 = $t79 + 111
100 addi $t81, $t80, 111 # $t81 = $t80 + 111
101 addi $t82, $t81, 111 # $t82 = $t81 + 111
102 addi $t83, $t82, 111 # $t83 = $t82 + 111
103 addi $t84, $t83, 111 # $t84 = $t83 + 111
104 addi $t85, $t84, 111 # $t85 = $t84 + 111
105 addi $t86, $t85, 111 # $t86 = $t85 + 111
106 addi $t87, $t86, 111 # $t87 = $t86 + 111
107 addi $t88, $t87, 111 # $t88 = $t87 + 111
108 addi $t89, $t88, 111 # $t89 = $t88 + 111
109 addi $t90, $t89, 111 # $t90 = $t89 + 111
110 addi $t91, $t90, 111 # $t91 = $t90 + 111
111 addi $t92, $t91, 111 # $t92 = $t91 + 111
112 addi $t93, $t92, 111 # $t93 = $t92 + 111
113 addi $t94, $t93, 111 # $t94 = $t93 + 111
114 addi $t95, $t94, 111 # $t95 = $t94 + 111
115 addi $t96, $t95, 111 # $t96 = $t95 + 111
116 addi $t97, $t96, 111 # $t97 = $t96 + 111
117 addi $t98, $t97, 111 # $t98 = $t97 + 111
118 addi $t99, $t98, 111 # $t99 = $t98 + 111
119 addi $t100, $t99, 111 # $t100 = $t99 + 111
120 addi $t101, $t100, 111 # $t101 = $t100 + 111
121 addi $t102, $t101, 111 # $t102 = $t101 + 111
122 addi $t103, $t102, 111 # $t103 = $t102 + 111
123 addi $t104, $t103, 111 # $t104 = $t103 + 111
124 addi $t105, $t104, 111 # $t105 = $t104 + 111
125 addi $t106, $t105, 111 # $t106 = $t105 + 111
126 addi $t107, $t106, 111 # $t107 = $t106 + 111
127 addi $t108, $t107, 111 # $t108 = $t107 + 111
128 addi $t109, $t108, 111 # $t109 = $t108 + 111
129 addi $t110, $t109, 111 # $t110 = $t109 + 111
130 addi $t111, $t110, 111 # $t111 = $t110 + 111
131 addi $t112, $t111, 111 # $t112 = $t111 + 111
132 addi $t113, $t112, 111 # $t113 = $t112 + 111
133 addi $t114, $t113, 111 # $t114 = $t113 + 111
134 addi $t115, $t114, 111 # $t115 = $t114 + 111
135 addi $t116, $t115, 111 # $t116 = $t115 + 111
136 addi $t117, $t116, 111 # $t117 = $t116 + 111
137 addi $t118, $t117, 111 # $t118 = $t117 + 111
138 addi $t119, $t118, 111 # $t119 = $t118 + 111
139 addi $t120, $t119, 111 # $t120 = $t119 + 111
140 addi $t121, $t120, 111 # $t121 = $t120 + 111
141 addi $t122, $t121, 111 # $t122 = $t121 + 111
142 addi $t123, $t122, 111 # $t123 = $t122 + 111
143 addi $t124, $t123, 111 # $t124 = $t123 + 111
144 addi $t125, $t124, 111 # $t125 = $t124 + 111
145 addi $t126, $t125, 111 # $t126 = $t125 + 111
146 addi $t127, $t126, 111 # $t127 = $t126 + 111
147 addi $t128, $t127, 111 # $t128 = $t127 + 111
148 addi $t129, $t128, 111 # $t129 = $t128 + 111
149 addi $t130, $t129, 111 # $t130 = $t129 + 111
150 addi $t131, $t130, 111 # $t131 = $t130 + 111
151 addi $t132, $t131, 111 # $t132 = $t131 + 111
152 addi $t133, $t132, 111 # $t133 = $t132 + 111
153 addi $t134, $t133, 111 # $t134 = $t133 + 111
154 addi $t135, $t134, 111 # $t135 = $t134 + 111
155 addi $t136, $t135, 111 # $t136 = $t135 + 111
156 addi $t137, $t136, 111 # $t137 = $t136 + 111
157 addi $t138, $t137, 111 # $t138 = $t137 + 111
158 addi $t139, $t138, 111 # $t139 = $t138 + 111
159 addi $t140, $t139, 111 # $t140 = $t139 + 111
160 addi $t141, $t140, 111 # $t141 = $t140 + 111
161 addi $t142, $t141, 111 # $t142 = $t141 + 111
162 addi $t143, $t142, 111 # $t143 = $t142 + 111
163 addi $t144, $t143, 111 # $t144 = $t143 + 111
164 addi $t145, $t144, 111 # $t145 = $t144 + 111
165 addi $t146, $t145, 111 # $t146 = $t145 + 111
166 addi $t147, $t146, 111 # $t147 = $t146 + 111
167 addi $t148, $t147, 111 # $t148 = $t147 + 111
168 addi $t149, $t148, 111 # $t149 = $t148 + 111
169 addi $t150, $t149, 111 # $t150 = $t149 + 111
170 addi $t151, $t150, 111 # $t151 = $t150 + 111
171 addi $t152, $t151, 111 # $t152 = $t151 + 111
172 addi $t153, $t152, 111 # $t153 = $t152 + 111
173 addi $t154, $t153, 111 # $t154 = $t153 + 111
174 addi $t155, $t154, 111 # $t155 = $t154 + 111
175 addi $t156, $t155, 111 # $t156 = $t155 + 111
176 addi $t157, $t156, 111 # $t157 = $t156 + 111
177 addi $t158, $t157, 111 # $t158 = $t157 + 111
178 addi $t159, $t158, 111 # $t159 = $t158 + 111
179 addi $t160, $t159, 111 # $t160 = $t159 + 111
180 addi $t161, $t160, 111 # $t161 = $t160 + 111
181 addi $t162, $t161, 111 # $t162 = $t161 + 111
182 addi $t163, $t162, 111 # $t163 = $t162 + 111
183 addi $t164, $t163, 111 # $t164 = $t163 + 111
184 addi $t165, $t164, 111 # $t165 = $t164 + 111
185 addi $t166, $t165, 111 # $t166 = $t165 + 111
186 addi $t167, $t166, 111 # $t167 = $t166 + 111
187 addi $t168, $t167, 111 # $t168 = $t167 + 111
188 addi $t169, $t168, 111 # $t169 = $t168 + 111
189 addi $t170, $t169, 111 # $t170 = $t169 + 111
190 addi $t171, $t170, 111 # $t171 = $t170 + 111
191 addi $t172, $t171, 111 # $t172 = $t171 + 111
192 addi $t173, $t172, 111 # $t173 = $t172 + 111
193 addi $t174, $t173, 111 # $t174 = $t173 + 111
194 addi $t175, $t174, 111 # $t175 = $t174 + 111
195 addi $t176, $t175, 111 # $t176 = $t175 + 111
196 addi $t177, $t176, 111 # $t177 = $t176 + 111
197 addi $t178, $t177, 111 # $t178 = $t177 + 111
198 addi $t179, $t178, 111 # $t179 = $t178 + 111
199 addi $t180, $t179, 111 # $t180 = $t179 + 111
200 addi $t181, $t180, 111 # $t181 = $t180 + 111
201 addi $t182, $t181, 111 # $t182 = $t181 + 111
202 addi $t183, $t182, 111 # $t183 = $t182 + 111
203 addi $t184, $t183, 111 # $t184 = $t183 + 111
204 addi $t185, $t184, 111 # $t185 = $t184 + 111
205 addi $t186, $t185, 111 # $t186 = $t185 + 111
206 addi $t187, $t186, 111 # $t187 = $t186 + 111
207 addi $t188, $t187, 111 # $t188 = $t187 + 111
208 addi $t189, $t188, 111 # $t189 = $t188 + 111
209 addi $t190, $t189, 111 # $t190 = $t189 + 111
210 addi $t191, $t190, 111 # $t191 = $t190 + 111
211 addi $t192, $t191, 111 # $t192 = $t191 + 111
212 addi $t193, $t192, 111 # $t193 = $t192 + 111
213 addi $t194, $t193, 111 # $t194 = $t193 + 111
214 addi $t195, $t194, 111 # $t195 = $t194 + 111
215 addi $t196, $t195, 111 # $t196 = $t195 + 111
216 addi $t197, $t196, 111 # $t197 = $t196 + 111
217 addi $t198, $t197, 111 # $t198 = $t197 + 111
218 addi $t199, $t198, 111 # $t199 = $t198 + 111
220 addi $t200, $t199, 111 # $t200 = $t199 + 111
221 addi $t201, $t200, 111 # $t201 = $t200 + 111
222 addi $t202, $t201, 111 # $t202 = $t201 + 111
223 addi $t203, $t202, 111 # $t203 = $t202 + 111
224 addi $t204, $t203, 111 # $t204 = $t203 + 111
225 addi $t205, $t204, 111 # $t205 = $t204 + 111
226 addi $t206, $t205, 111 # $t206 = $t205 + 111
227 addi $t207, $t206, 111 # $t207 = $t206 + 111
228 addi $t208, $t207, 111 # $t208 = $t207 + 111
229 addi $t209, $t208, 111 # $t209 = $t208 + 111
230 addi $t210, $t209, 111 # $t210 = $t209 + 111
231 addi $t211, $t210, 111 # $t211 = $t210 + 111
232 addi $t212, $t211, 111 # $t212 = $t211 + 111
233 addi $t213, $t212, 111 # $t213 = $t212 + 111
234 addi $t214, $t213, 111 # $t214 = $t213 + 111
235 addi $t215, $t214, 111 # $t215 = $t214 + 111
236 addi $t216, $t215, 111 # $t216 = $t215 + 111
237 addi $t217, $t216, 111 # $t217 = $t216 + 111
238 addi $t218, $t217, 111 # $t218 = $t217 + 111
239 addi $t219, $t218, 111 # $t219 = $t218 + 111
240 addi $t220, $t219, 111 # $t220 = $t219 + 111
241 addi $t221, $t220, 111 # $t221 = $t220 + 111
242 addi $t222, $t221, 111 # $t222 = $t221 + 111
243 addi $t223, $t222, 111 # $t223 = $t222 + 111
244 addi $t224, $t223, 111 # $t224 = $t223 + 111
245 addi $t225, $t224, 111 # $t225 = $t224 + 111
246 addi $t226, $t225, 111 # $t226 = $t225 + 111
247 addi $t227, $t226, 111 # $t227 = $t226 + 111
248 addi $t228, $t227, 111 # $t228 = $t227 + 111
249 addi $t229, $t228, 111 # $t229 = $t228 + 111
250 addi $t230, $t229, 111 # $t230 = $t229 + 111
251 addi $t231, $t230, 111 # $t231 = $t230 + 111
252 addi $t232, $t231, 111 # $t232 = $t231 + 111
253 addi $t233, $t232, 111 # $t233 = $t232 + 111
254 addi $t234, $t233, 111 # $t234 = $t233 + 111
255 addi $t235, $t234, 111 # $t235 = $t234 + 111
256 addi $t236, $t235, 111 # $t236 = $t235 + 111
257 addi $t237, $t236, 111 # $t237 = $t236 + 111
258 addi $t238, $t237, 111 # $t238 = $t237 + 111
259 addi $t239, $t238, 111 # $t239 = $t238 + 111
260 addi $t240, $t239, 111 # $t240 = $t239 + 111
261 addi $t241, $t240, 111 # $t241 = $t240 + 111
262 addi $t242, $t241, 111 # $t242 = $t241 + 111
263 addi $t243, $t242, 111 # $t243 = $t242 + 111
264 addi $t244, $t243, 111 # $t244 = $t243 + 111
265 addi $t245, $t244, 111 # $t245 = $t244 + 111
266 addi $t246, $t245, 111 # $t246 = $t245 + 111
267 addi $t247, $t246, 111 # $t247 = $t246 + 111
268 addi $t248, $t247, 111 # $t248 = $t247 + 111
269 addi $t249, $t248, 111 # $t249 = $t248 + 111
270 addi $t250, $t249, 111 # $t250 = $t249 + 111
271 addi $t251, $t250, 111 # $t251 = $t250 + 111
272 addi $t252, $t251, 111 # $t252 = $t251 + 111
273 addi $t253, $t252, 111 # $t253 = $t252 + 111
274 addi $t254, $t253, 111 # $t254 = $t253 + 111
275 addi $t255, $t254, 111 # $t255 = $t254 + 111
276 addi $t256, $t255, 111 # $t256 = $t255 + 111
277 addi $t257, $t256, 111 # $t257 = $t256 + 111
278 addi $t258, $t257, 111 # $t258 = $t257 + 111
279 addi $t259, $t258, 111 # $t259 = $t258 + 111
280 addi $t260, $t259, 111 # $t260 = $t259 + 111
281 addi $t261, $t260, 111 # $t261 = $t260 + 111
282 addi $t262, $t261, 111 # $t262 = $t261 + 111
283 addi $t263, $t262, 111 # $t263 = $t262 + 111
284 addi $t264, $t263, 111 # $t264 = $t263 + 111
285 addi $t265, $t264, 111 # $t265 = $t264 + 111
286 addi $t266, $t265, 111 # $t266 = $t265 + 111
287 addi $t267, $t266, 111 # $t267 = $t266 + 111
288 addi $t268, $t267, 111 # $t268 = $t267 + 111
289 addi $t269, $t268, 111 # $t269 = $t268 + 111
290 addi $t270, $t269, 111 # $t270 = $t269 + 111
291 addi $t271, $t270, 111 # $t271 = $t270 + 111
292 addi $t272, $t271, 111 # $t272 = $t271 + 111
293 addi $t273, $t272, 111 # $t273 = $t272 + 111
294 addi $t274, $t273, 111 # $t274 = $t273 + 111
295 addi $t275, $t274, 111 # $t275 = $t274 + 111
296 addi $t276, $t275, 111 # $t276 = $t275 + 111
297 addi $t277, $t276, 111 # $t277 = $t276 + 111
298 addi $t278, $t277, 111 # $t278 = $t277 + 111
299 addi $t279, $t278, 111 # $t279 = $t278 + 111
300 addi $t280, $t279, 111 # $t280 = $t279 + 111
301 addi $t281, $t280, 111 # $t281 = $t280 + 111
302 addi $t282, $t281, 111 # $t282 = $t281 + 111
303 addi $t283, $t282, 111 # $t283 = $t282 + 111
304 addi $t284, $t283, 111 # $t284 = $t283 + 111
305 addi $t285, $t284, 111 # $t285 = $t284 + 111
306 addi $t286, $t285, 111 # $t286 = $t285 + 111
307 addi $t287, $t286, 111 # $t287 = $t286 + 111
308 addi $t288, $t287, 111 # $t288 = $t287 + 111
309 addi $t289, $t288, 111 # $t289 = $t288 + 111
310 addi $t290, $t289, 111 # $t290 = $t289 + 111
311 addi $t291, $t290, 111 # $t291 = $t290 + 111
312 addi $t292, $t291, 111 # $t292 = $t291 + 111
313 addi $t293, $t292, 111 # $t293 = $t292 + 111
314 addi $t294, $t293, 111 # $t294 = $t293 + 111
315 addi $t295, $t294, 111 # $t295 = $t294 + 111
316 addi $t296, $t295, 111 # $t296 = $t295 + 111
317 addi $t297, $t296, 111 # $t297 = $t296 + 111
318 addi $t298, $t297, 111 # $t298 = $t297 + 111
319 addi $t299, $t298, 111 # $t299 = $t298 + 111
320 addi $t300, $t299, 111 # $t300 = $t299 + 111
321 addi $t301, $t300, 111 # $t301 = $t300 + 111
322 addi $t302, $t301, 111 # $t302 = $t301 + 111
323 addi $t303, $t302, 111 # $t303 = $t302 + 111
324 addi $t304, $t303, 111 # $t304 = $t303 + 111
325 addi $t305, $t304, 111 # $t305 = $t304 + 111
326 addi $t306, $t305, 111 # $t306 = $t305 + 111
327 addi $t307, $t306, 111 # $t307 = $t306 + 111
328 addi $t308, $t307, 111 # $t308 = $t307 + 111
329 addi $t309, $t308, 111 # $t309 = $t308 + 111
330 addi $t310, $t309, 111 # $t310 = $t309 + 111
331 addi $t311, $t310, 111 # $t311 = $t310 + 111
332 addi $t312, $t311, 111 # $t312 = $t311 + 111
333 addi $t313, $t312, 111 # $t313 = $t312 + 111
334 addi $t314, $t313, 111 # $t314 = $t313 + 111
335 addi $t315, $t314, 111 # $t315 = $t314 + 111
336 addi $t316, $t315, 111 # $t316 = $t315 + 111
337 addi $t317, $t316, 111 # $t317 = $t316 + 111
338 addi $t318, $t317, 111 # $t318 = $t317 + 111
339 addi $t319, $t318, 111 # $t319 = $t318 + 111
340 addi $t320, $t319, 111 # $t320 = $t319 + 111
341 addi $t321, $t320, 111 # $t321 = $t320 + 111
342 addi $t322, $t321, 111 # $t322 = $t321 + 111
343 addi $t323, $t322, 111 # $t323 = $t322 + 111
344 addi $t324, $t323, 111 # $t324 = $t323 + 111
345 addi $t325, $t324, 111 # $t325 = $t3
```

# MIPS Commands

The screenshot shows a MIPS assembly debugger interface. The assembly code window contains the following instructions:

```
1 # Shows 'Hello, world!' at the top of the stack
2 .text
3 .globl _start
4 .type _start, @function
5 _start:
6    li $t0, 115902
7    li $t1, 115901
8    li $t2, 115900
9    li $t3, 115901
10   li $t4, 115900
11   li $t5, 115901
12   li $t6, 115900
13   li $t7, 115901
14   li $t8, 115900
15   li $t9, 115901
16   li $t10, 115900
17   li $t11, 115901
18   li $t12, 115900
19   li $t13, 115901
20   li $t14, 115900
21   li $t15, 115901
22   li $t16, 115900
23   li $t17, 115901
24   li $t18, 115900
25   li $t19, 115901
26   li $t20, 115900
27   li $t21, 115901
28   li $t22, 115900
29   li $t23, 115901
30   li $t24, 115900
31   li $t25, 115901
32   li $t26, 115900
33   li $t27, 115901
34   li $t28, 115900
35   li $t29, 115901
36   li $t30, 115900
37   li $t31, 115901
38   li $t32, 115900
39   li $t33, 115901
40   li $t34, 115900
41   li $t35, 115901
42   li $t36, 115900
43   li $t37, 115901
44   li $t38, 115900
45   li $t39, 115901
46   li $t40, 115900
47   li $t41, 115901
48   li $t42, 115900
49   li $t43, 115901
50   li $t44, 115900
51   li $t45, 115901
52   li $t46, 115900
53   li $t47, 115901
54   li $t48, 115900
55   li $t49, 115901
56   li $t50, 115900
57   li $t51, 115901
58   li $t52, 115900
59   li $t53, 115901
60   li $t54, 115900
61   li $t55, 115901
62   li $t56, 115900
63   li $t57, 115901
64   li $t58, 115900
65   li $t59, 115901
66   li $t60, 115900
67   li $t61, 115901
68   li $t62, 115900
69   li $t63, 115901
70   li $t64, 115900
71   li $t65, 115901
72   li $t66, 115900
73   li $t67, 115901
74   li $t68, 115900
75   li $t69, 115901
76   li $t70, 115900
77   li $t71, 115901
78   li $t72, 115900
79   li $t73, 115901
80   li $t74, 115900
81   li $t75, 115901
82   li $t76, 115900
83   li $t77, 115901
84   li $t78, 115900
85   li $t79, 115901
86   li $t80, 115900
87   li $t81, 115901
88   li $t82, 115900
89   li $t83, 115901
90   li $t84, 115900
91   li $t85, 115901
92   li $t86, 115900
93   li $t87, 115901
94   li $t88, 115900
95   li $t89, 115901
96   li $t90, 115900
97   li $t91, 115901
98   li $t92, 115900
99   li $t93, 115901
100  li $t94, 115900
101  li $t95, 115901
102  li $t96, 115900
103  li $t97, 115901
104  li $t98, 115900
105  li $t99, 115901
106  li $t100, 115900
107  li $t101, 115901
108  li $t102, 115900
109  li $t103, 115901
110  li $t104, 115900
111  li $t105, 115901
112  li $t106, 115900
113  li $t107, 115901
114  li $t108, 115900
115  li $t109, 115901
116  li $t110, 115900
117  li $t111, 115901
118  li $t112, 115900
119  li $t113, 115901
120  li $t114, 115900
121  li $t115, 115901
122  li $t116, 115900
123  li $t117, 115901
124  li $t118, 115900
125  li $t119, 115901
126  li $t120, 115900
127  li $t121, 115901
128  li $t122, 115900
129  li $t123, 115901
130  li $t124, 115900
131  li $t125, 115901
132  li $t126, 115900
133  li $t127, 115901
134  li $t128, 115900
135  li $t129, 115901
136  li $t130, 115900
137  li $t131, 115901
138  li $t132, 115900
139  li $t133, 115901
140  li $t134, 115900
141  li $t135, 115901
142  li $t136, 115900
143  li $t137, 115901
144  li $t138, 115900
145  li $t139, 115901
146  li $t140, 115900
147  li $t141, 115901
148  li $t142, 115900
149  li $t143, 115901
150  li $t144, 115900
151  li $t145, 115901
152  li $t146, 115900
153  li $t147, 115901
154  li $t148, 115900
155  li $t149, 115901
156  li $t150, 115900
157  li $t151, 115901
158  li $t152, 115900
159  li $t153, 115901
160  li $t154, 115900
161  li $t155, 115901
162  li $t156, 115900
163  li $t157, 115901
164  li $t158, 115900
165  li $t159, 115901
166  li $t160, 115900
167  li $t161, 115901
168  li $t162, 115900
169  li $t163, 115901
170  li $t164, 115900
171  li $t165, 115901
172  li $t166, 115900
173  li $t167, 115901
174  li $t168, 115900
175  li $t169, 115901
176  li $t170, 115900
177  li $t171, 115901
178  li $t172, 115900
179  li $t173, 115901
180  li $t174, 115900
181  li $t175, 115901
182  li $t176, 115900
183  li $t177, 115901
184  li $t178, 115900
185  li $t179, 115901
186  li $t180, 115900
187  li $t181, 115901
188  li $t182, 115900
189  li $t183, 115901
190  li $t184, 115900
191  li $t185, 115901
192  li $t186, 115900
193  li $t187, 115901
194  li $t188, 115900
195  li $t189, 115901
196  li $t190, 115900
197  li $t191, 115901
198  li $t192, 115900
199  li $t193, 115901
200  li $t194, 115900
201  li $t195, 115901
202  li $t196, 115900
203  li $t197, 115901
204  li $t198, 115900
205  li $t199, 115901
206  li $t200, 115900
207  li $t201, 115901
208  li $t202, 115900
209  li $t203, 115901
210  li $t204, 115900
211  li $t205, 115901
212  li $t206, 115900
213  li $t207, 115901
214  li $t208, 115900
215  li $t209, 115901
216  li $t210, 115900
217  li $t211, 115901
218  li $t212, 115900
219  li $t213, 115901
220  li $t214, 115900
221  li $t215, 115901
222  li $t216, 115900
223  li $t217, 115901
224  li $t218, 115900
225  li $t219, 115901
226  li $t220, 115900
227  li $t221, 115901
228  li $t222, 115900
229  li $t223, 115901
230  li $t224, 115900
231  li $t225, 115901
232  li $t226, 115900
233  li $t227, 115901
234  li $t228, 115900
235  li $t229, 115901
236  li $t230, 115900
237  li $t231, 115901
238  li $t232, 115900
239  li $t233, 115901
240  li $t234, 115900
241  li $t235, 115901
242  li $t236, 115900
243  li $t237, 115901
244  li $t238, 115900
245  li $t239, 115901
246  li $t240, 115900
247  li $t241, 115901
248  li $t242, 115900
249  li $t243, 115901
250  li $t244, 115900
251  li $t245, 115901
252  li $t246, 115900
253  li $t247, 115901
254  li $t248, 115900
255  li $t249, 115901
256  li $t250, 115900
257  li $t251, 115901
258  li $t252, 115900
259  li $t253, 115901
260  li $t254, 115900
261  li $t255, 115901
262  li $t256, 115900
263  li $t257, 115901
264  li $t258, 115900
265  li $t259, 115901
266  li $t260, 115900
267  li $t261, 115901
268  li $t262, 115900
269  li $t263, 115901
270  li $t264, 115900
271  li $t265, 115901
272  li $t266, 115900
273  li $t267, 115901
274  li $t268, 115900
275  li $t269, 115901
276  li $t270, 115900
277  li $t271, 115901
278  li $t272, 115900
279  li $t273, 115901
280  li $t274, 115900
281  li $t275, 115901
282  li $t276, 115900
283  li $t277, 115901
284  li $t278, 115900
285  li $t279, 115901
286  li $t280, 115900
287  li $t281, 115901
288  li $t282, 115900
289  li $t283, 115901
290  li $t284, 115900
291  li $t285, 115901
292  li $t286, 115900
293  li $t287, 115901
294  li $t288, 115900
295  li $t289, 115901
296  li $t290, 115900
297  li $t291, 115901
298  li $t292, 115900
299  li $t293, 115901
300  li $t294, 115900
301  li $t295, 115901
302  li $t296, 115900
303  li $t297, 115901
304  li $t298, 115900
305  li $t299, 115901
306  li $t300, 115900
307  li $t301, 115901
308  li $t302, 115900
309  li $t303, 115901
310  li $t304, 115900
311  li $t305, 115901
312  li $t306, 115900
313  li $t307, 115901
314  li $t308, 115900
315  li $t309, 115901
316  li $t310, 115900
317  li $t311, 115901
318  li $t312, 115900
319  li $t313, 115901
320  li $t314, 115900
321  li $t315, 115901
322  li $t316, 115900
323  li $t317, 115901
324  li $t318, 115900
325  li $t319, 115901
326  li $t320, 115900
327  li $t321, 115901
328  li $t322, 115900
329  li $t323, 115901
330  li $t324, 115900
331  li $t325, 115901
332  li $t326, 115900
333  li $t327, 115901
334  li $t328, 115900
335  li $t329, 115901
336  li $t330, 115900
337  li $t331, 115901
338  li $t332, 115900
339  li $t333, 115901
340  li $t334, 115900
341  li $t335, 115901
342  li $t336, 115900
343  li $t337, 115901
344  li $t338, 115900
345  li $t339, 115901
346  li $t340, 115900
347  li $t341, 115901
348  li $t342, 115900
349  li $t343, 115901
350  li $t344, 115900
351  li $t345, 115901
352  li $t346, 115900
353  li $t347, 115901
354  li $t348, 115900
355  li $t349, 115901
356  li $t350, 115900
357  li $t351, 115901
358  li $t352, 115900
359  li $t353, 115901
360  li $t354, 115900
361  li $t355, 115901
362  li $t356, 115900
363  li $t357, 115901
364  li $t358, 115900
365  li $t359, 115901
366  li $t360, 115900
367  li $t361, 115901
368  li $t362, 115900
369  li $t363, 115901
370  li $t364, 115900
371  li $t365, 115901
372  li $t366, 115900
373  li $t367, 115901
374  li $t368, 115900
375  li $t369, 115901
376  li $t370, 115900
377  li $t371, 115901
378  li $t372, 115900
379  li $t373, 115901
380  li $t374, 115900
381  li $t375, 115901
382  li $t376, 115900
383  li $t377, 115901
384  li $t378, 115900
385  li $t379, 115901
386  li $t380, 115900
387  li $t381, 115901
388  li $t382, 115900
389  li $t383, 115901
390  li $t384, 115900
391  li $t385, 115901
392  li $t386, 115900
393  li $t387, 115901
394  li $t388, 115900
395  li $t389, 115901
396  li $t390, 115900
397  li $t391, 115901
398  li $t392, 115900
399  li $t393, 115901
400  li $t394, 115900
401  li $t395, 115901
402  li $t396, 115900
403  li $t397, 115901
404  li $t398, 115900
405  li $t399, 115901
406  li $t400, 115900
407  li $t401, 115901
408  li $t402, 115900
409  li $t403, 115901
410  li $t404, 115900
411  li $t405, 115901
412  li $t406, 115900
413  li $t407, 115901
414  li $t408, 115900
415  li $t409, 115901
416  li $t410, 115900
417  li $t411, 115901
418  li $t412, 115900
419  li $t413, 115901
420  li $t414, 115900
421  li $t415, 115901
422  li $t416, 115900
423  li $t417, 115901
424  li $t418, 115900
425  li $t419, 115901
426  li $t420, 115900
427  li $t421, 115901
428  li $t422, 115900
429  li $t423, 115901
430  li $t424, 115900
431  li $t425, 115901
432  li $t426, 115900
433  li $t427, 115901
434  li $t428, 115900
435  li $t429, 115901
436  li $t430, 115900
437  li $t431, 115901
438  li $t432, 115900
439  li $t433, 115901
440  li $t434, 115900
441  li $t435, 115901
442  li $t436, 115900
443  li $t437, 115901
444  li $t438, 115900
445  li $t439, 115901
446  li $t440, 115900
447  li $t441, 115901
448  li $t442, 115900
449  li $t443, 115901
450  li $t444, 115900
451  li $t445, 115901
452  li $t446, 115900
453  li $t447, 115901
454  li $t448, 115900
455  li $t449, 115901
456  li $t450, 115900
457  li $t451, 115901
458  li $t452, 115900
459  li $t453, 115901
460  li $t454, 115900
461  li $t455, 115901
462  li $t456, 115900
463  li $t457, 115901
464  li $t458, 115900
465  li $t459, 115901
466  li $t460, 115900
467  li $t461, 115901
468  li $t462, 115900
469  li $t463, 115901
470  li $t464, 115900
471  li $t465, 115901
472  li $t466, 115900
473  li $t467, 115901
474  li $t468, 115900
475  li $t469, 115901
476  li $t470, 115900
477  li $t471, 115901
478  li $t472, 115900
479  li $t473, 115901
480  li $t474, 115900
481  li $t475, 115901
482  li $t476, 115900
483  li $t477, 115901
484  li $t478, 115900
485  li $t479, 115901
486  li $t480, 115900
487  li $t481, 115901
488  li $t482, 115900
489  li $t483, 115901
490  li $t484, 115900
491  li $t485, 115901
492  li $t486, 115900
493  li $t487, 115901
494  li $t488, 115900
495  li $t489, 115901
496  li $t490, 115900
497  li $t491, 115901
498  li $t492, 115900
499  li $t493, 115901
500  li $t494, 115900
501  li $t495, 115901
502  li $t496, 115900
503  li $t497, 115901
504  li $t498, 115900
505  li $t499, 115901
506  li $t500, 115900
507  li $t501, 115901
508  li $t502, 115900
509  li $t503, 115901
510  li $t504, 115900
511  li $t505, 115901
512  li $t506, 115900
513  li $t507, 115901
514  li $t508, 115900
515  li $t509, 115901
516  li $t510, 115900
517  li $t511, 115901
518  li $t512, 115900
519  li $t513, 115901
520  li $t514, 115900
521  li $t515, 115901
522  li $t516, 115900
523  li $t517, 115901
524  li $t518, 115900
525  li $t519, 115901
526  li $t520, 115900
527  li $t521, 115901
528  li $t522, 115900
529  li $t523, 115901
530  li $t524, 115900
531  li $t525, 115901
532  li $t526, 115900
533  li $t527, 115901
534  li $t528, 115900
535  li $t529, 115901
536  li $t530, 115900
537  li $t531, 115901
538  li $t532, 115900
539  li $t533, 115901
540  li $t534, 115900
541  li $t535, 115901
542  li $t536, 115900
543  li $t537, 115901
544  li $t538, 115900
545  li $t539, 115901
546  li $t540, 115900
547  li $t541, 115901
548  li $t542, 115900
549  li $t543, 115901
550  li $t544, 115900
551  li $t545, 115901
552  li $t546, 115900
553  li $t547, 115901
554  li $t548, 115900
555  li $t549, 115901
556  li $t550, 115900
557  li $t551, 115901
558  li $t552, 115900
559  li $t553, 115901
560  li $t554, 115900
561  li $t555, 115901
562  li $t556, 115900
563  li $t557, 115901
564  li $t558, 115900
565  li $t559, 115901
566  li $t560, 115900
567  li $t561, 115901
568  li $t562, 115900
569  li $t563, 115901
570  li $t564, 115900
571  li $t565, 115901
572  li $t566, 115900
573  li $t567, 115901
574  li $t568, 115900
575  li $t569, 115901
576  li $t570, 115900
577  li $t571, 115901
578  li $t572, 115900
579  li $t573, 115901
580  li $t574, 115900
581  li $t575, 115901
582  li $t576, 115900
583  li $t577, 115901
584  li $t578, 115900
585  li $t579, 115901
586  li $t580, 115900
587  li $t581, 115901
588  li $t582, 115900
589  li $t583, 115901
590  li $t584, 115900
591  li $t585, 115901
592  li $t586, 115900
593  li $t587, 115901
594  li $t588, 115900
595  li $t589, 115901
596  li $t590, 115900
597  li $t591, 115901
598  li $t592, 115900
599  li $t593, 115901
600  li $t594, 115900
601  li $t595, 115901
602  li $t596, 115900
603  li $t597, 115901
604  li $t598, 115900
605  li $t599, 115901
606  li $t600, 115900
607  li $t601, 115901
608  li $t602, 115900
609  li $t603, 115901
610  li $t604, 115900
611  li $t605, 115901
612  li $t606, 115900
613  li $t607, 115901
614  li $t608, 115900
615  li $t609, 115901
616  li $t610, 115900
617  li $t611, 115901
618  li $t612, 115900
619  li $t613, 115901
620  li $t614, 115900
621  li $t615, 115901
622  li $t616, 115900
623  li $t617, 115901
624  li $t618, 115900
625  li $t619, 115901
626  li $t620, 115900
627  li $t621, 115901
628  li $t622, 115900
629  li $t623, 115901
630  li $t624, 115900
631  li $t625, 115901
632  li $t626, 115900
633  li $t627, 115901
634  li $t628, 115900
635  li $t629, 115901
636  li $t630, 115900
637  li $t631, 115901
638  li $t632, 115900
639  li $t633, 115901
640  li $t634, 115900
641  li $t635, 115901
642  li $t636, 115900
643  li $t637, 115901
644  li $t638, 115900
645  li $t639, 115901
646  li $t640, 115900
647  li $t641, 115901
648  li $t642, 115900
649  li $t643, 115901
650  li $t644, 115900
651  li $t645, 115901
652  li $t646, 115900
653  li $t647, 115901
654  li $t648, 115900
655  li $t649, 115901
656  li $t650, 115900
657  li $t651, 115901
658  li $t652, 115900
659  li $t653, 115901
660  li $t654, 115900
661  li $
```

# MIPS Commands

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

```
1 # Shows 'Hello world' at the top of the stack
2 .text
3 .globl _start
4 .data
5 _start: .asciz "Hello world"
6 .text
7 _start: li $t0, 111901
8 sb $t0, 111901
9 li $t1, 111902
10 sb $t1, 111902
11 li $t2, 111903
12 sb $t2, 111903
13 li $t3, 111904
14 sb $t3, 111904
15 li $t4, 111905
16 sb $t4, 111905
17 li $t5, 111906
18 sb $t5, 111906
19 li $t6, 111907
20 sb $t6, 111907
21 li $t7, 111908
22 sb $t7, 111908
23 li $t8, 111909
24 sb $t8, 111909
25 li $t9, 111910
26 sb $t9, 111910
27 li $t10, 111911
28 sb $t10, 111911
29 li $t11, 111912
30 sb $t11, 111912
31 li $t12, 111913
32 sb $t12, 111913
33 li $t13, 111914
34 sb $t13, 111914
35 li $t14, 111915
36 sb $t14, 111915
37 li $t15, 111916
38 sb $t15, 111916
39 li $t16, 111917
40 sb $t16, 111917
41 li $t17, 111918
42 sb $t17, 111918
43 li $t18, 111919
44 sb $t18, 111919
45 li $t19, 111920
46 sb $t19, 111920
47 li $t20, 111921
48 sb $t20, 111921
49 li $t21, 111922
50 sb $t21, 111922
51 li $t22, 111923
52 sb $t22, 111923
53 li $t23, 111924
54 sb $t23, 111924
55 li $t24, 111925
56 sb $t24, 111925
57 li $t25, 111926
58 sb $t25, 111926
59 li $t26, 111927
60 sb $t26, 111927
61 li $t27, 111928
62 sb $t27, 111928
63 li $t28, 111929
64 sb $t28, 111929
65 li $t29, 111930
66 sb $t29, 111930
67 li $t30, 111931
68 sb $t30, 111931
69 li $t31, 111932
70 sb $t31, 111932
71 li $t32, 111933
72 sb $t32, 111933
73 li $t33, 111934
74 sb $t33, 111934
75 li $t34, 111935
76 sb $t34, 111935
77 li $t35, 111936
78 sb $t35, 111936
79 li $t36, 111937
80 sb $t36, 111937
81 li $t37, 111938
82 sb $t37, 111938
83 li $t38, 111939
84 sb $t38, 111939
85 li $t39, 111940
86 sb $t39, 111940
87 li $t40, 111941
88 sb $t40, 111941
89 li $t41, 111942
90 sb $t41, 111942
91 li $t42, 111943
92 sb $t42, 111943
93 li $t43, 111944
94 sb $t43, 111944
95 li $t44, 111945
96 sb $t44, 111945
97 li $t45, 111946
98 sb $t45, 111946
99 li $t46, 111947
100 sb $t46, 111947
101 li $t47, 111948
102 sb $t47, 111948
103 li $t48, 111949
104 sb $t48, 111949
105 li $t49, 111950
106 sb $t49, 111950
107 li $t50, 111951
108 sb $t50, 111951
109 li $t51, 111952
110 sb $t51, 111952
111 li $t52, 111953
112 sb $t52, 111953
113 li $t53, 111954
114 sb $t53, 111954
115 li $t54, 111955
116 sb $t54, 111955
117 li $t55, 111956
118 sb $t55, 111956
119 li $t56, 111957
120 sb $t56, 111957
121 li $t57, 111958
122 sb $t57, 111958
123 li $t58, 111959
124 sb $t58, 111959
125 li $t59, 111960
126 sb $t59, 111960
127 li $t60, 111961
128 sb $t60, 111961
129 li $t61, 111962
130 sb $t61, 111962
131 li $t62, 111963
132 sb $t62, 111963
133 li $t63, 111964
134 sb $t63, 111964
135 li $t64, 111965
136 sb $t64, 111965
137 li $t65, 111966
138 sb $t65, 111966
139 li $t66, 111967
140 sb $t66, 111967
141 li $t67, 111968
142 sb $t67, 111968
143 li $t68, 111969
144 sb $t68, 111969
145 li $t69, 111970
146 sb $t69, 111970
147 li $t70, 111971
148 sb $t70, 111971
149 li $t71, 111972
150 sb $t71, 111972
151 li $t72, 111973
152 sb $t72, 111973
153 li $t73, 111974
154 sb $t73, 111974
155 li $t74, 111975
156 sb $t74, 111975
157 li $t75, 111976
158 sb $t75, 111976
159 li $t76, 111977
160 sb $t76, 111977
161 li $t77, 111978
162 sb $t77, 111978
163 li $t78, 111979
164 sb $t78, 111979
165 li $t79, 111980
166 sb $t79, 111980
167 li $t80, 111981
168 sb $t80, 111981
169 li $t81, 111982
170 sb $t81, 111982
171 li $t82, 111983
172 sb $t82, 111983
173 li $t83, 111984
174 sb $t83, 111984
175 li $t84, 111985
176 sb $t84, 111985
177 li $t85, 111986
178 sb $t85, 111986
179 li $t86, 111987
180 sb $t86, 111987
181 li $t87, 111988
182 sb $t87, 111988
183 li $t88, 111989
184 sb $t88, 111989
185 li $t89, 111990
186 sb $t89, 111990
187 li $t90, 111991
188 sb $t90, 111991
189 li $t91, 111992
190 sb $t91, 111992
191 li $t92, 111993
192 sb $t92, 111993
193 li $t93, 111994
194 sb $t93, 111994
195 li $t94, 111995
196 sb $t94, 111995
197 li $t95, 111996
198 sb $t95, 111996
199 li $t96, 111997
200 sb $t96, 111997
201 li $t97, 111998
202 sb $t97, 111998
203 li $t98, 111999
204 sb $t98, 111999
205 li $t99, 111900
206 sb $t99, 111900
207 li $t100, 111901
208 sb $t100, 111901
209 li $t101, 111902
210 sb $t101, 111902
211 li $t102, 111903
212 sb $t102, 111903
213 li $t103, 111904
214 sb $t103, 111904
215 li $t104, 111905
216 sb $t104, 111905
217 li $t105, 111906
218 sb $t105, 111906
219 li $t106, 111907
220 sb $t106, 111907
221 li $t107, 111908
222 sb $t107, 111908
223 li $t108, 111909
224 sb $t108, 111909
225 li $t109, 111910
226 sb $t109, 111910
227 li $t110, 111911
228 sb $t110, 111911
229 li $t111, 111912
230 sb $t111, 111912
231 li $t112, 111913
232 sb $t112, 111913
233 li $t113, 111914
234 sb $t113, 111914
235 li $t114, 111915
236 sb $t114, 111915
237 li $t115, 111916
238 sb $t115, 111916
239 li $t116, 111917
240 sb $t116, 111917
241 li $t117, 111918
242 sb $t117, 111918
243 li $t118, 111919
244 sb $t118, 111919
245 li $t119, 111920
246 sb $t119, 111920
247 li $t120, 111921
248 sb $t120, 111921
249 li $t121, 111922
250 sb $t121, 111922
251 li $t122, 111923
252 sb $t122, 111923
253 li $t123, 111924
254 sb $t123, 111924
255 li $t124, 111925
256 sb $t124, 111925
257 li $t125, 111926
258 sb $t125, 111926
259 li $t126, 111927
260 sb $t126, 111927
261 li $t127, 111928
262 sb $t127, 111928
263 li $t128, 111929
264 sb $t128, 111929
265 li $t129, 111930
266 sb $t129, 111930
267 li $t130, 111931
268 sb $t130, 111931
269 li $t131, 111932
270 sb $t131, 111932
271 li $t132, 111933
272 sb $t132, 111933
273 li $t133, 111934
274 sb $t133, 111934
275 li $t134, 111935
276 sb $t134, 111935
277 li $t135, 111936
278 sb $t135, 111936
279 li $t136, 111937
280 sb $t136, 111937
281 li $t137, 111938
282 sb $t137, 111938
283 li $t138, 111939
284 sb $t138, 111939
285 li $t139, 111940
286 sb $t139, 111940
287 li $t140, 111941
288 sb $t140, 111941
289 li $t141, 111942
290 sb $t141, 111942
291 li $t142, 111943
292 sb $t142, 111943
293 li $t143, 111944
294 sb $t143, 111944
295 li $t144, 111945
296 sb $t144, 111945
297 li $t145, 111946
298 sb $t145, 111946
299 li $t146, 111947
300 sb $t146, 111947
301 li $t147, 111948
302 sb $t147, 111948
303 li $t148, 111949
304 sb $t148, 111949
305 li $t149, 111950
306 sb $t149, 111950
307 li $t150, 111951
308 sb $t150, 111951
309 li $t151, 111952
310 sb $t151, 111952
311 li $t152, 111953
312 sb $t152, 111953
313 li $t153, 111954
314 sb $t153, 111954
315 li $t154, 111955
316 sb $t154, 111955
317 li $t155, 111956
318 sb $t155, 111956
319 li $t156, 111957
320 sb $t156, 111957
321 li $t157, 111958
322 sb $t157, 111958
323 li $t158, 111959
324 sb $t158, 111959
325 li $t159, 111960
326 sb $t159, 111960
327 li $t160, 111961
328 sb $t160, 111961
329 li $t161, 111962
330 sb $t161, 111962
331 li $t162, 111963
332 sb $t162, 111963
333 li $t163, 111964
334 sb $t163, 111964
335 li $t164, 111965
336 sb $t164, 111965
337 li $t165, 111966
338 sb $t165, 111966
339 li $t166, 111967
340 sb $t166, 111967
341 li $t167, 111968
342 sb $t167, 111968
343 li $t168, 111969
344 sb $t168, 111969
345 li $t169, 111970
346 sb $t169, 111970
347 li $t170, 111971
348 sb $t170, 111971
349 li $t171, 111972
350 sb $t171, 111972
351 li $t172, 111973
352 sb $t172, 111973
353 li $t173, 111974
354 sb $t173, 111974
355 li $t174, 111975
356 sb $t174, 111975
357 li $t175, 111976
358 sb $t175, 111976
359 li $t176, 111977
360 sb $t176, 111977
361 li $t177, 111978
362 sb $t177, 111978
363 li $t178, 111979
364 sb $t178, 111979
365 li $t179, 111980
366 sb $t179, 111980
367 li $t180, 111981
368 sb $t180, 111981
369 li $t181, 111982
370 sb $t181, 111982
371 li $t182, 111983
372 sb $t182, 111983
373 li $t183, 111984
374 sb $t183, 111984
375 li $t184, 111985
376 sb $t184, 111985
377 li $t185, 111986
378 sb $t185, 111986
379 li $t186, 111987
380 sb $t186, 111987
381 li $t187, 111988
382 sb $t187, 111988
383 li $t188, 111989
384 sb $t188, 111989
385 li $t189, 111990
386 sb $t189, 111990
387 li $t190, 111991
388 sb $t190, 111991
389 li $t191, 111992
390 sb $t191, 111992
391 li $t192, 111993
392 sb $t192, 111993
393 li $t193, 111994
394 sb $t193, 111994
395 li $t194, 111995
396 sb $t194, 111995
397 li $t195, 111996
398 sb $t195, 111996
399 li $t196, 111997
400 sb $t196, 111997
401 li $t197, 111998
402 sb $t197, 111998
403 li $t198, 111999
404 sb $t198, 111999
405 li $t199, 111900
406 sb $t199, 111900
407 li $t200, 111901
408 sb $t200, 111901
409 li $t201, 111902
410 sb $t201, 111902
411 li $t202, 111903
412 sb $t202, 111903
413 li $t203, 111904
414 sb $t203, 111904
415 li $t204, 111905
416 sb $t204, 111905
417 li $t205, 111906
418 sb $t205, 111906
419 li $t206, 111907
420 sb $t206, 111907
421 li $t207, 111908
422 sb $t207, 111908
423 li $t208, 111909
424 sb $t208, 111909
425 li $t209, 111910
426 sb $t209, 111910
427 li $t210, 111911
428 sb $t210, 111911
429 li $t211, 111912
430 sb $t211, 111912
431 li $t212, 111913
432 sb $t212, 111913
433 li $t213, 111914
434 sb $t213, 111914
435 li $t214, 111915
436 sb $t214, 111915
437 li $t215, 111916
438 sb $t215, 111916
439 li $t216, 111917
440 sb $t216, 111917
441 li $t217, 111918
442 sb $t217, 111918
443 li $t218, 111919
444 sb $t218, 111919
445 li $t219, 111920
446 sb $t219, 111920
447 li $t220, 111921
448 sb $t220, 111921
449 li $t221, 111922
450 sb $t221, 111922
451 li $t222, 111923
452 sb $t222, 111923
453 li $t223, 111924
454 sb $t223, 111924
455 li $t224, 111925
456 sb $t224, 111925
457 li $t225, 111926
458 sb $t225, 111926
459 li $t226, 111927
460 sb $t226, 111927
461 li $t227, 111928
462 sb $t227, 111928
463 li $t228, 111929
464 sb $t228, 111929
465 li $t229, 111930
466 sb $t229, 111930
467 li $t230, 111931
468 sb $t230, 111931
469 li $t231, 111932
470 sb $t231, 111932
471 li $t232, 111933
472 sb $t232, 111933
473 li $t233, 111934
474 sb $t233, 111934
475 li $t234, 111935
476 sb $t234, 111935
477 li $t235, 111936
478 sb $t235, 111936
479 li $t236, 111937
480 sb $t236, 111937
481 li $t237, 111938
482 sb $t237, 111938
483 li $t238, 111939
484 sb $t238, 111939
485 li $t239, 111940
486 sb $t239, 111940
487 li $t240, 111941
488 sb $t240, 111941
489 li $t241, 111942
490 sb $t241, 111942
491 li $t242, 111943
492 sb $t242, 111943
493 li $t243, 111944
494 sb $t243, 111944
495 li $t244, 111945
496 sb $t244, 111945
497 li $t245, 111946
498 sb $t245, 111946
499 li $t246, 111947
500 sb $t246, 111947
501 li $t247, 111948
502 sb $t247, 111948
503 li $t248, 111949
504 sb $t248, 111949
505 li $t249, 111950
506 sb $t249, 111950
507 li $t250, 111951
508 sb $t250, 111951
509 li $t251, 111952
510 sb $t251, 111952
511 li $t252, 111953
512 sb $t252, 111953
513 li $t253, 111954
514 sb $t253, 111954
515 li $t254, 111955
516 sb $t254, 111955
517 li $t255, 111956
518 sb $t255, 111956
519 li $t256, 111957
520 sb $t256, 111957
521 li $t257, 111958
522 sb $t257, 111958
523 li $t258, 111959
524 sb $t258, 111959
525 li $t259, 111960
526 sb $t259, 111960
527 li $t260, 111961
528 sb $t260, 111961
529 li $t261, 111962
530 sb $t261, 111962
531 li $t262, 111963
532 sb $t262, 111963
533 li $t263, 111964
534 sb $t263, 111964
535 li $t264, 111965
536 sb $t264, 111965
537 li $t265, 111966
538 sb $t265, 111966
539 li $t266, 111967
540 sb $t266, 111967
541 li $t267, 111968
542 sb $t267, 111968
543 li $t268, 111969
544 sb $t268, 111969
545 li $t269, 111970
546 sb $t269, 111970
547 li $t270, 111971
548 sb $t270, 111971
549 li $t271, 111972
550 sb $t271, 111972
551 li $t272, 111973
552 sb $t272, 111973
553 li $t273, 111974
554 sb $t273, 111974
555 li $t274, 111975
556 sb $t274, 111975
557 li $t275, 111976
558 sb $t275, 111976
559 li $t276, 111977
560 sb $t276, 111977
561 li $t277, 111978
562 sb $t277, 111978
563 li $t278, 111979
564 sb $t278, 111979
565 li $t279, 111980
566 sb $t279, 111980
567 li $t280, 111981
568 sb $t280, 111981
569 li $t281, 111982
570 sb $t281, 111982
571 li $t282, 111983
572 sb $t282, 111983
573 li $t283, 111984
574 sb $t283, 111984
575 li $t284, 111985
576 sb $t284, 111985
577 li $t285, 111986
578 sb $t285, 111986
579 li $t286, 111987
580 sb $t286, 111987
581 li $t287, 111988
582 sb $t287, 111988
583 li $t288, 111989
584 sb $t288, 111989
585 li $t289, 111990
586 sb $t289, 111990
587 li $t290, 111991
588 sb $t290, 111991
589 li $t291, 111992
590 sb $t291, 111992
591 li $t292, 111993
592 sb $t292, 111993
593 li $t293, 111994
594 sb $t293, 111994
595 li $t294, 111995
596 sb $t294, 111995
597 li $t295, 111996
598 sb $t295, 111996
599 li $t296, 111997
600 sb $t296, 111997
601 li $t297, 111998
602 sb $t297, 111998
603 li $t298, 111999
604 sb $t298, 111999
605 li $t299, 111900
606 sb $t299, 111900
607 li $t300, 111901
608 sb $t300, 111901
609 li $t301, 111902
610 sb $t301, 111902
611 li $t302, 111903
612 sb $t302, 111903
613 li $t303, 111904
614 sb $t303, 111904
615 li $t304, 111905
616 sb $t304, 111905
617 li $t305, 111906
618 sb $t305, 111906
619 li $t306, 111907
620 sb $t306, 111907
621 li $t307, 111908
622 sb $t307, 111908
623 li $t308, 111909
624 sb $t308, 111909
625 li $t309, 111910
626 sb $t309, 111910
627 li $t310, 111911
628 sb $t310, 111911
629 li $t311, 111912
630 sb $t311, 111912
631 li $t312, 111913
632 sb $t312, 111913
633 li $t313, 111914
634 sb $t313, 111914
635 li $t314, 111915
636 sb $t314, 111915
637 li $t315, 111916
638 sb $t315, 111916
639 li $t316, 111917
640 sb $t316, 111917
641 li $t317, 111918
642 sb $t317, 111918
643 li $t318, 111919
644 sb $t318, 111919
645 li $t319, 111920
646 sb $t319, 111920
647 li $t320, 111921
648 sb $t320, 111921
649 li $t321, 111922
650 sb $t321, 111922
651 li $t322, 111923
652 sb $t322, 111923
653 li $t323, 111924
654 sb $t323, 111924
655 li $t324, 111925
656 sb $t324, 111925
657 li $t325, 111926
658 sb $t325, 111926
659 li $t326, 111927
660 sb $t326, 111927
661 li $t327, 111928
662 sb $t327, 111928
663 li $t328, 111929
664 sb $t328, 111929
665 li $t329, 111930
666 sb $t329, 111930
667 li $t330, 111931
668 sb $t330, 111931
669 li $t331, 111932
670 sb $t331, 111932
671 li $t332, 111933
672 sb $t332, 111933
673 li $t333, 111934
674 sb $t333, 111934
675 li $t334, 111935
676 sb $t334, 111935
677 li $t335, 111936
678 sb $t335, 111936
679 li $t336, 111937
680 sb $t336, 111937
681 li $t337, 111938
682 sb $t337, 111938
683 li $t338, 111939
684 sb $t338, 111939
685 li $t339, 111940
686 sb $t339, 111940
687 li $t340, 111941
688 sb $t340, 111941
689 li $t341, 111942
690 sb $t341, 111942
691 li $t342, 111943
692 sb $t342, 111943
693 li $t343, 111944
694 sb $t343, 111944
695 li $t344, 111945
696 sb $t344, 111945
697 li $t345, 111946
698 sb $t345, 111946
699 li $t346, 1119
```

# Challenge:

Line: 3 Go! Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

Step Run  Enable auto switching

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

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

# WeMIPS

The screenshot shows the WeMIPS IDE interface. At the top, there are tabs for 'Run', '3', 'Data', and 'ShowHide Device'. Below these are navigation tabs: 'Addition Doubler', 'Stax', 'Looper', 'Stack Test', and 'Hello World'. Underneath 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    ADDI $t0,$zero,72 # N
2    ADDI $t1,$zero,101 # e
3    ADDI $t2,$zero,101 # m
4    ADDI $t3,$zero,101 # l
5    ADDI $t4,$zero,101 # o
6    ADDI $t5,$zero,101 # r
7    ADDI $t6,$zero,101 # i
8    ADDI $t7,$zero,101 # n
9    ADDI $t8,$zero,101 # d
10   ADDI $t9,$zero,33 # !
11   ADDI $t10,$zero,0 # null
12   ADDI $t11,$zero,4 # 4 is for print string
13   ADDI $t12,$zero,0 # point to the log
14
15   # syscall
```

To the right, a debugger window titled 'Step' shows registers T, A, V, Stack, and Log. The registers are as follows:

Reg	T	A	V	Stack	Log
s0	10				
s1	9				
s2	8				
s3	22				
s4	695				
s5	976				
s6	977				
s7	419				

(Demo with WeMIPS)

# Today's Topics



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

# Loops & Jumps in Machine Language

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



# Loops & Jumps in Machine Language

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



# Loops & Jumps in Machine Language

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



# Loops & Jumps in Machine Language

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



# Loops & Jumps in Machine Language

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



# Loops & Jumps in Machine Language

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



# Loops & Jumps in Machine Language

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



# Jump Demo

Line: 18 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

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

(Demo  
with  
WeMIPS)

Step Run  Enable auto switching

S T A V Stack Log

Emulation complete, returning to line 1

abcdefghijklmnopqrstuvwxyz

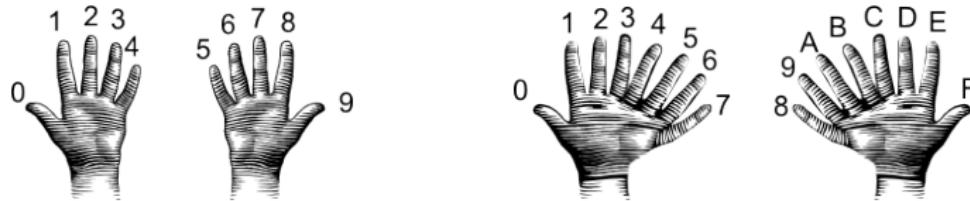


# Today's Topics



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

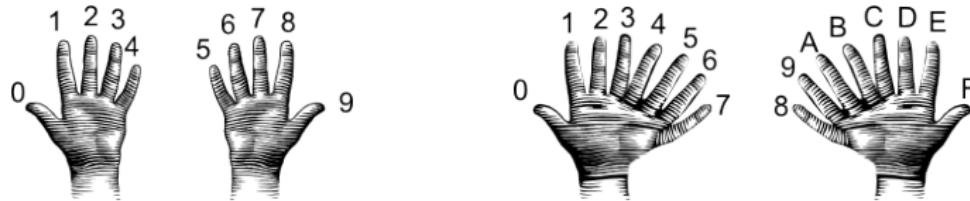
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

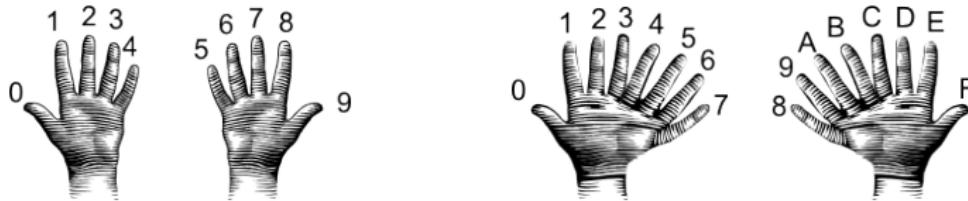
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

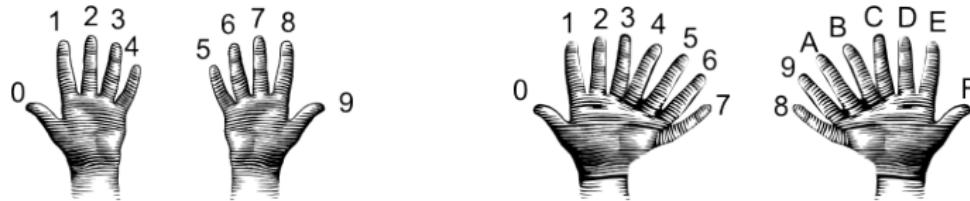
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

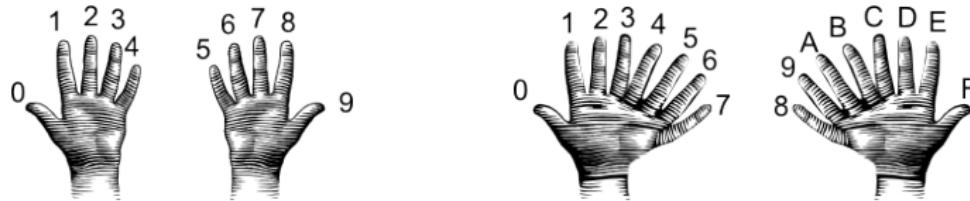
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

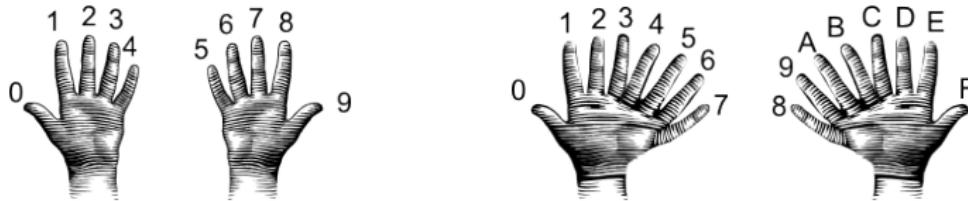
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

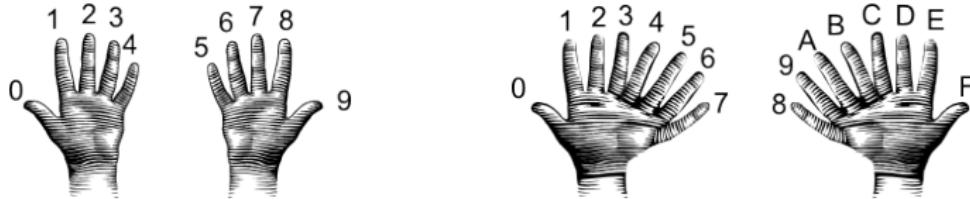
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

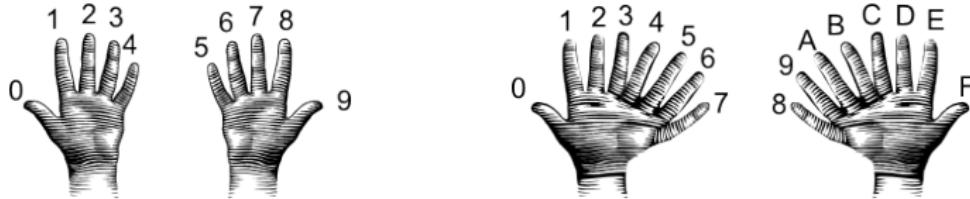
- Convert first digit to decimal and multiple by 16.
  - Convert second digit to decimal and add to total.
  - Example: what is 2A as a decimal number?

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

A in decimal digits is 10.

$32 + 10$  is 42.

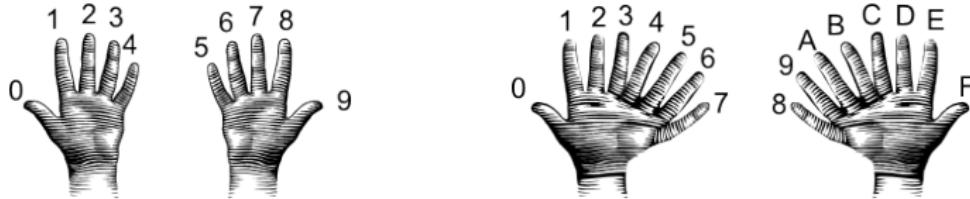
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
  - ▶ Convert first digit to decimal and multiple by 16.
  - ▶ Convert second digit to decimal and add to total.
  - ▶ Example: what is 2A as a decimal number?  
2 in decimal is 2.  $2 \times 16$  is 32.  
A in decimal digits is 10.  
 $32 + 10$  is 42.  
Answer is 42.
  - ▶ Example: what is 99 as a decimal number?

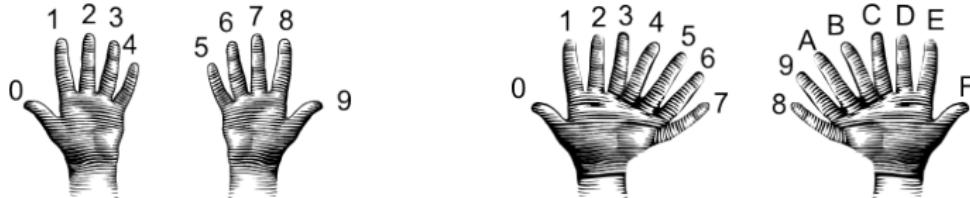
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
  - Convert first digit to decimal and multiple by 16.
  - Convert second digit to decimal and add to total.
  - Example: what is 2A as a decimal number?  
2 in decimal is 2.  $2 \times 16$  is 32.  
A in decimal digits is 10.  
 $32 + 10$  is 42.  
Answer is 42.
  - Example: what is 99 as a decimal number?  
9 in decimal is 9.

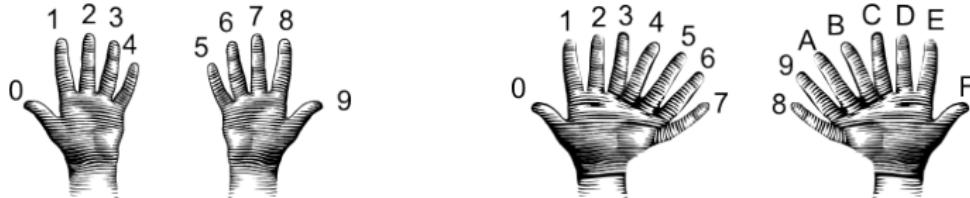
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
  - Convert first digit to decimal and multiple by 16.
  - Convert second digit to decimal and add to total.
  - Example: what is 2A as a decimal number?  
2 in decimal is 2.  $2 \times 16$  is 32.  
A in decimal digits is 10.  
 $32 + 10$  is 42.  
Answer is 42.
  - Example: what is 99 as a decimal number?  
9 in decimal is 9.  $9 \times 16$  is 144.

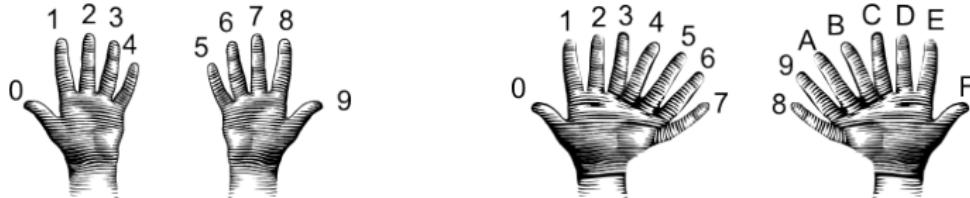
# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

- From hexadecimal to decimal (assuming two-digit numbers):
  - Convert first digit to decimal and multiple by 16.
  - Convert second digit to decimal and add to total.
  - Example: what is 2A as a decimal number?  
2 in decimal is 2.  $2 \times 16$  is 32.  
A in decimal digits is 10.  
 $32 + 10$  is 42.  
Answer is 42.
  - Example: what is 99 as a decimal number?  
9 in decimal is 9.  $9 \times 16$  is 144.  
9 in decimal digits is 9

# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

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

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

A in decimal digits is 10.

$32 + 10$  is 42.

Answer is 42.

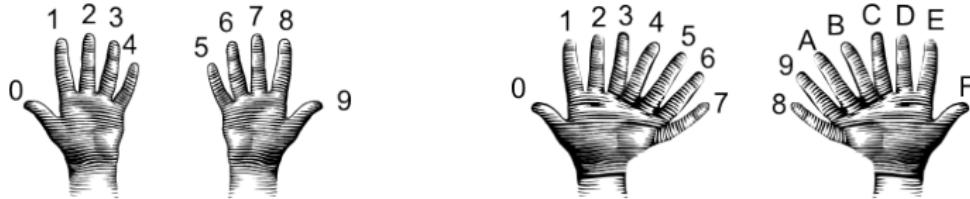
- Example: what is 99 as a decimal number?

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

9 in decimal digits is 9

$144 + 9$  is 153.

# Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

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

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

A in decimal digits is 10.

$32 + 10$  is 42.

Answer is 42.

- Example: what is 99 as a decimal number?

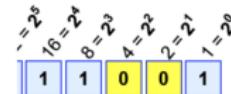
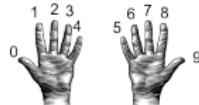
9 in decimal is 9.  $9 \times 16$  is 144.

9 in decimal digits is 9

$144 + 9$  is 153.

Answer is 153.

# Decimal to Binary: Converting Between Bases

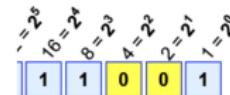
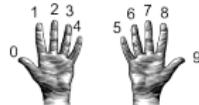


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

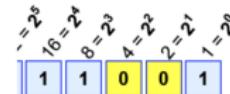


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

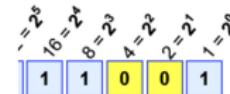
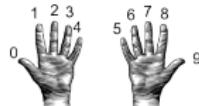


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

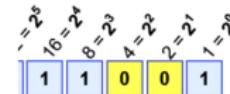


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

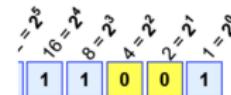
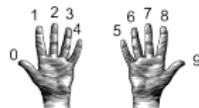


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

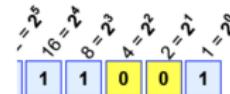
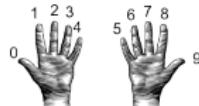


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

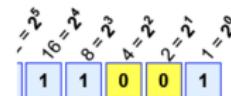
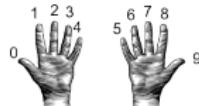


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

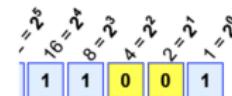
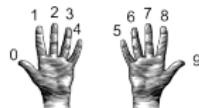


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases

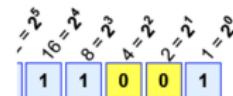
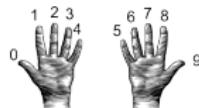


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

# Decimal to Binary: Converting Between Bases



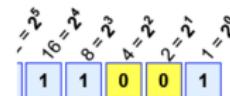
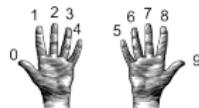
- From decimal to binary:

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

► Example: what is 130 in binary notation?

130/128 is 1 rem 2.

# Decimal to Binary: Converting Between Bases

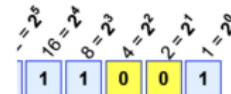
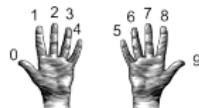


- From decimal to binary:

- 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



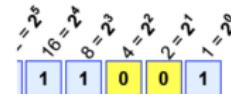
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

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

# Decimal to Binary: Converting Between Bases



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

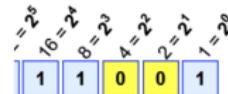
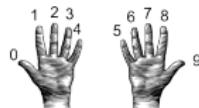
- From decimal to binary:

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

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

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

# Decimal to Binary: Converting Between Bases



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

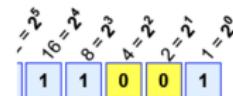
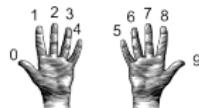
- From decimal to binary:

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

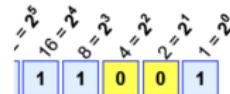
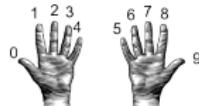
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

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

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

2/32 is 0 rem 2.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

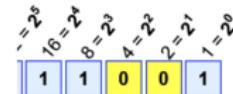
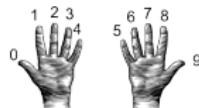
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

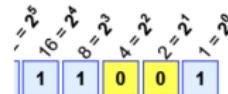
- Divide by  $128 (= 2^7)$ . Quotient is the first digit.
- Divide remainder by  $64 (= 2^6)$ . Quotient is the next digit.
- Divide remainder by  $32 (= 2^5)$ . Quotient is the next digit.
- Divide remainder by  $16 (= 2^4)$ . Quotient is the next digit.
- Divide remainder by  $8 (= 2^3)$ . Quotient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

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

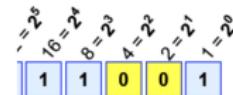
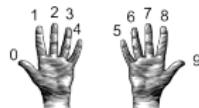
130/128 is 1 rem 2. First digit is 1: 1...

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

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

2/16 is 0 rem 2.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

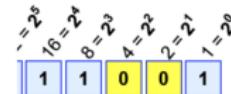
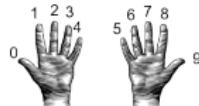
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

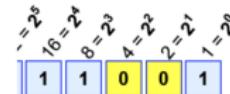
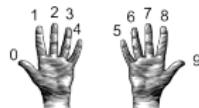
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

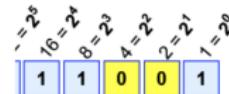
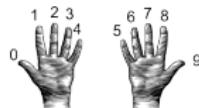
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

2/8 is 0 rem 2.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

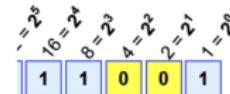
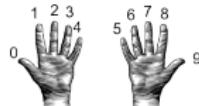
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

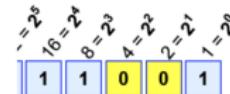
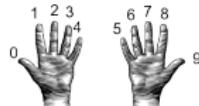
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

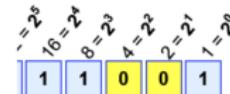
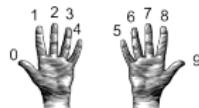
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

2/4 is 0 remainder 2.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

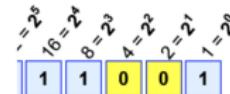
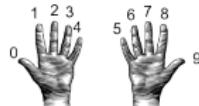
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

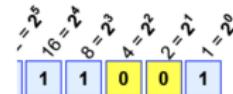
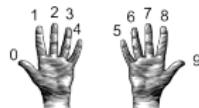
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

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

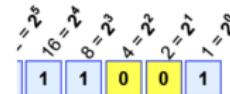
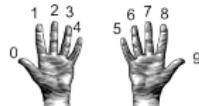
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

2/2 is 1 rem 0.

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

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

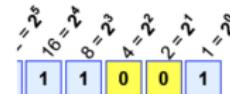
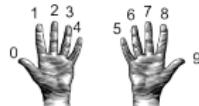
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

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

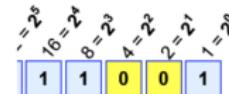
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

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

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

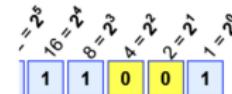
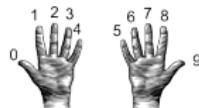
2/8 is 0 rem 2. Next digit is 0: 10000...

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

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

Adding the last remainder: 10000010

# Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

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

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

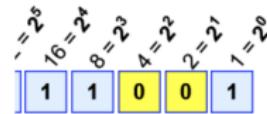
2/8 is 0 rem 2. Next digit is 0: 10000...

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

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

Adding the last remainder: 10000010

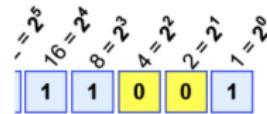
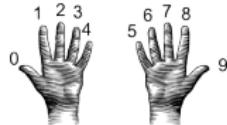
# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

# Decimal to Binary: Converting Between Bases

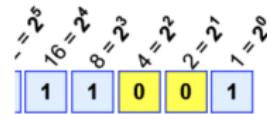


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99 / 128$  is 0 rem 99.

# Decimal to Binary: Converting Between Bases

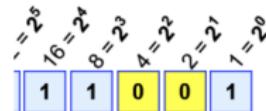
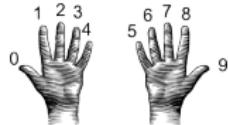


Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

- Example: what is 99 in binary notation?

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

# Decimal to Binary: Converting Between Bases



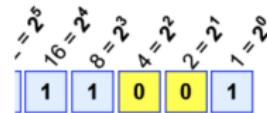
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

99/64 is 1 rem 35.

# Decimal to Binary: Converting Between Bases



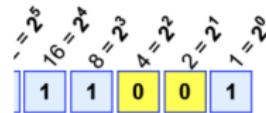
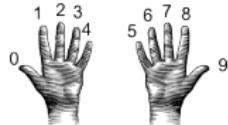
Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- Example: what is 99 in binary notation?

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

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

# Decimal to Binary: Converting Between Bases



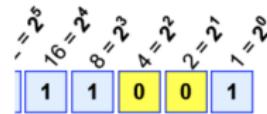
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

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

# Decimal to Binary: Converting Between Bases



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

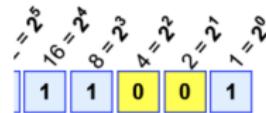
- Example: what is 99 in binary notation?

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

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

35/32 is 1 rem 3.

# Decimal to Binary: Converting Between Bases



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

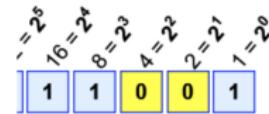
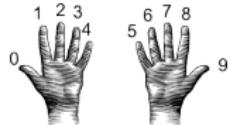
- Example: what is 99 in binary notation?

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

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

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

# Decimal to Binary: Converting Between Bases



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

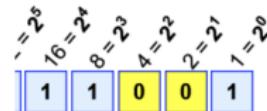
- Example: what is 99 in binary notation?

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

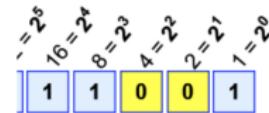
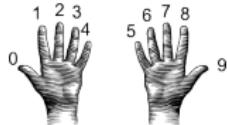
99/128 is 0 rem 99. First digit is 0: 0...

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

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

3/16 is 0 rem 3.

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

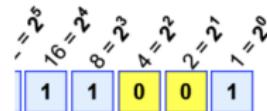
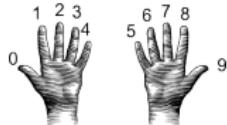
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

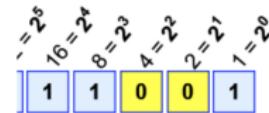
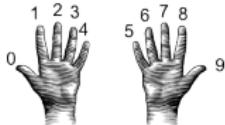
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

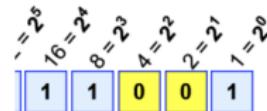
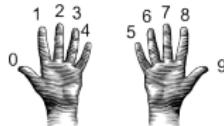
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

3/8 is 0 rem 3.

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

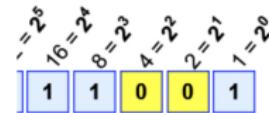
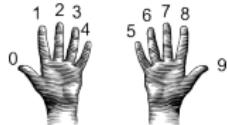
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

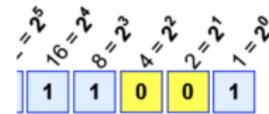
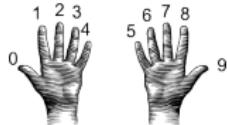
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

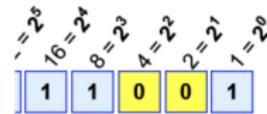
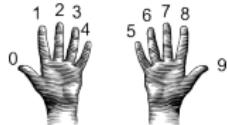
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

3/4 is 0 remainder 3.

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

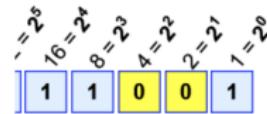
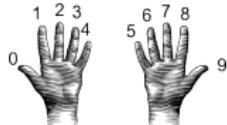
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

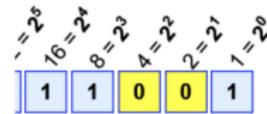
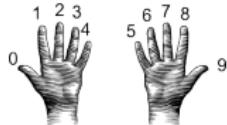
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

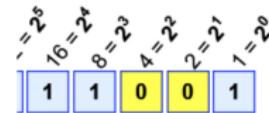
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1.

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

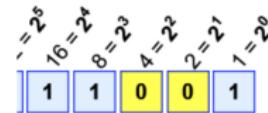
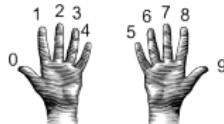
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

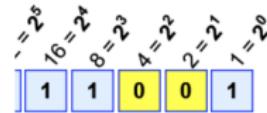
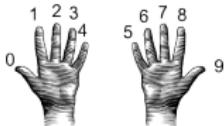
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

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

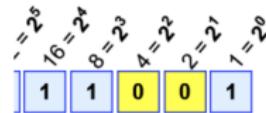
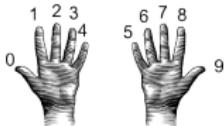
3/8 is 0 rem 3. Next digit is 0: 01100...

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

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

Adding the last remainder: 01100011

# Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

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

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

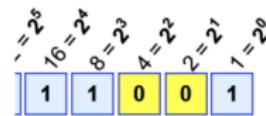
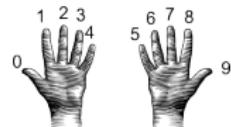
3/4 is 0 remainder 3. Next digit is 0: 011000...

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

Adding the last remainder: 01100011

Answer is 1100011.

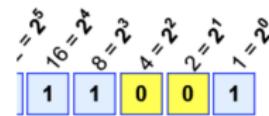
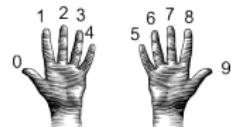
# Binary to Decimal: Converting Between Bases



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

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

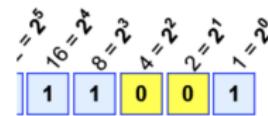
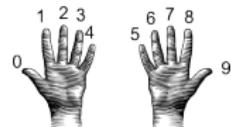
# Binary to Decimal: Converting Between Bases



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

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

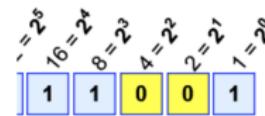
# Binary to Decimal: Converting Between Bases



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

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

# Binary to Decimal: Converting Between Bases

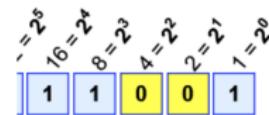
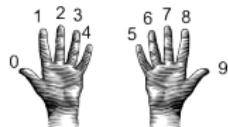


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

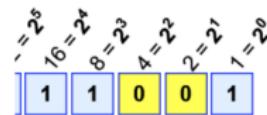
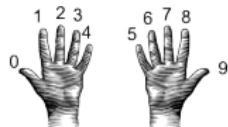


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

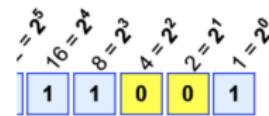
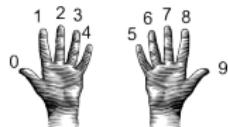


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

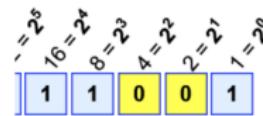
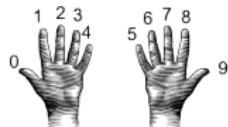


Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

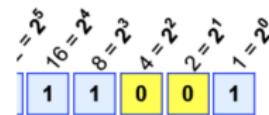
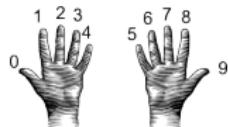


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases

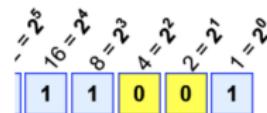
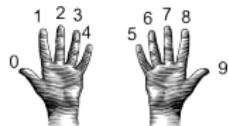


Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- From binary to decimal:

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

# Binary to Decimal: Converting Between Bases



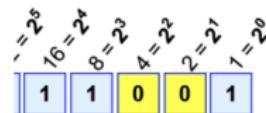
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Sum starts with:

# Binary to Decimal: Converting Between Bases



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

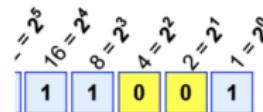
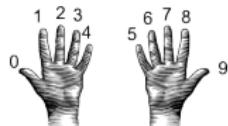
- From binary to decimal:

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

Sum starts with: 1

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

# Binary to Decimal: Converting Between Bases



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

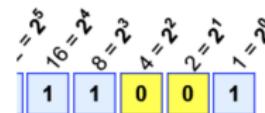
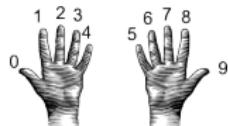
- From binary to decimal:

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

Sum starts with: 1

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

# Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

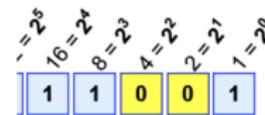
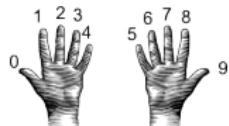
- Set sum = last digit.
- Multiply next digit by  $2^1$ . Add to sum.
- Multiply next digit by  $4 = 2^2$ . Add to sum.
- Multiply next digit by  $8 = 2^3$ . Add to sum.
- Multiply next digit by  $16 = 2^4$ . Add to sum.
- Multiply next digit by  $32 = 2^5$ . Add to sum.
- Multiply next digit by  $64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

Sum starts with: 1

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

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

# Binary to Decimal: Converting Between Bases



- From binary to decimal:

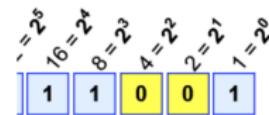
- Set sum = last digit.
- Multiply next digit by  $2 = 2^1$ . Add to sum.
- Multiply next digit by  $4 = 2^2$ . Add to sum.
- Multiply next digit by  $8 = 2^3$ . Add to sum.
- Multiply next digit by  $16 = 2^4$ . Add to sum.
- Multiply next digit by  $32 = 2^5$ . Add to sum.
- Multiply next digit by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



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

- From binary to decimal:

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

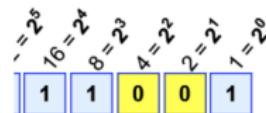
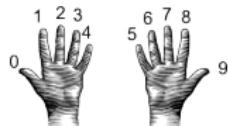
Sum starts with: 1

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

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

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

# Binary to Decimal: Converting Between Bases



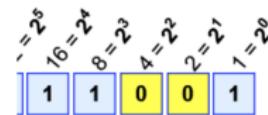
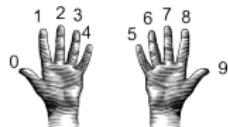
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



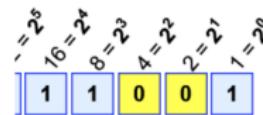
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



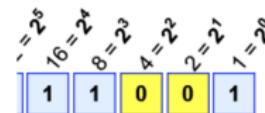
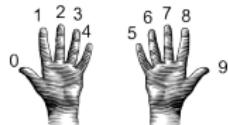
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



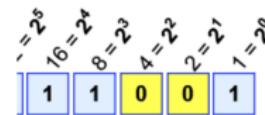
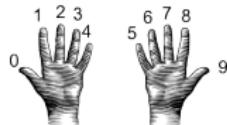
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



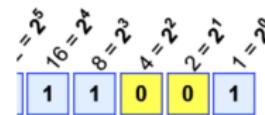
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases



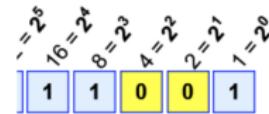
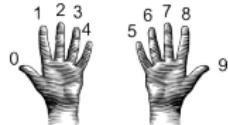
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

# Binary to Decimal: Converting Between Bases

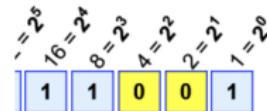
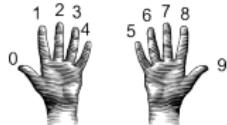


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

# Binary to Decimal: Converting Between Bases



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

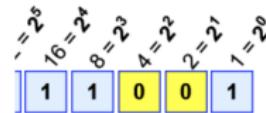
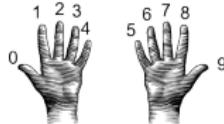
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

# Binary to Decimal: Converting Between Bases



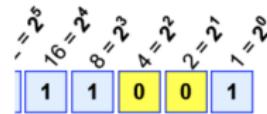
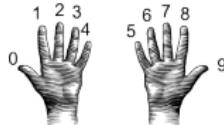
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

# Binary to Decimal: Converting Between Bases



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

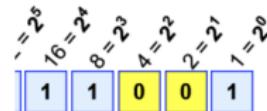
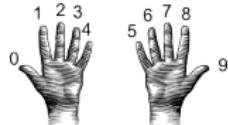
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

# Binary to Decimal: Converting Between Bases



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

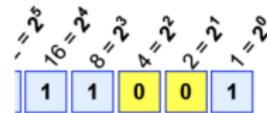
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

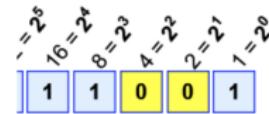
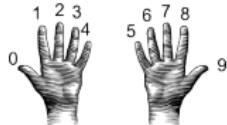
Sum starts with: 0

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

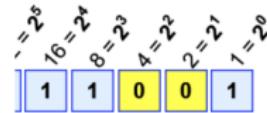
Sum starts with: 0

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

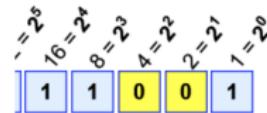
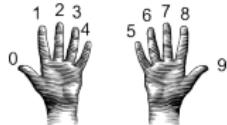
$0 \times 2 = 0$ . Add 0 to sum: 0

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

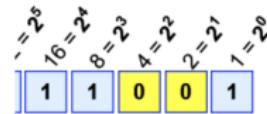
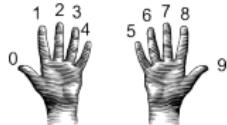
$0 \times 2 = 0$ . Add 0 to sum: 0

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

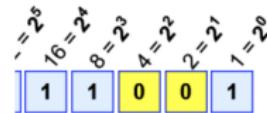
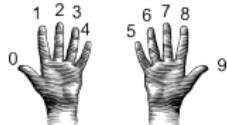
$1 \times 4 = 4$ . Add 4 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

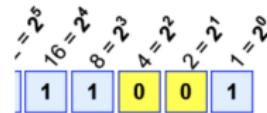
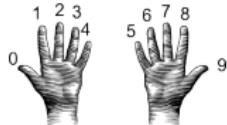
$1 \times 4 = 4$ . Add 4 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

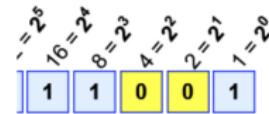
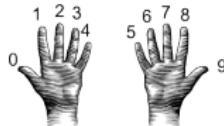
$0 \times 8 = 0$ . Add 0 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

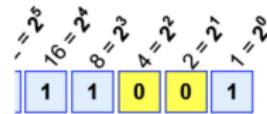
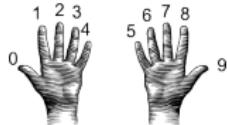
$0 \times 8 = 0$ . Add 0 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

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

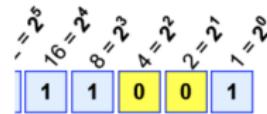
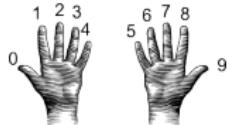
$0 \times 16 = 0$ . Add 0 to sum: 4

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

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

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

# Binary to Decimal: Converting Between Bases

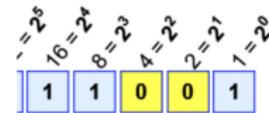
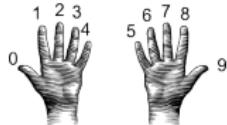


Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 = 16 + 8 + 4 = 25$

- Example: What is 10100100 in decimal?

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

# Binary to Decimal: Converting Between Bases



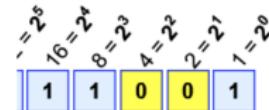
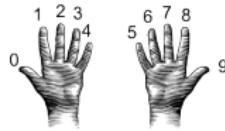
Example:  $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 = 16 + 8 + 4 = 25$

- Example: What is 10100100 in decimal?

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

The answer is 164.

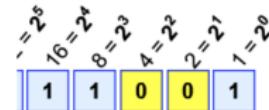
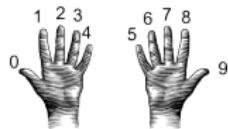
# Design Challenge: Incrementers



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

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

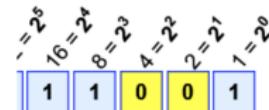
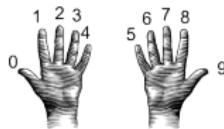
# Design Challenge: Incrementers



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

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

# Design Challenge: Incrementers

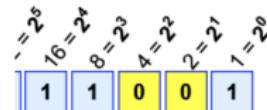
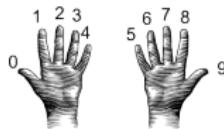


Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

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

# Design Challenge: Incrementers



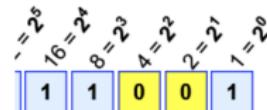
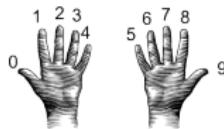
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

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

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

# Design Challenge: Incrementers



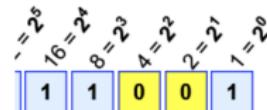
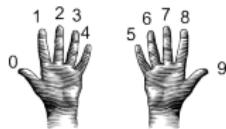
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

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

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

# Design Challenge: Incrementers



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

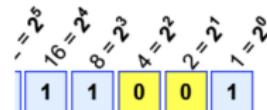
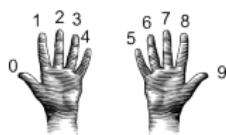
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

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

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

# Design Challenge: Incrementers



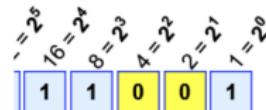
Example:  $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

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

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

# Design Challenge: Incrementers



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

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

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

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

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

- Challenge: Write an algorithm for incrementing binary numbers.

Example: "1001" → "1010"

# Recap

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



# Recap



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

# Recap



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

# Recap



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

## Final Overview: Format

- The exam is 2 hours long.

# Final Overview: Format

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

## Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Final Overview: Format

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

# Exam Options

## Exam Times:

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

11 December 2008

### Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commissing, attempting to commit or aiding in the commission of acts of academic dishonesty) as offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy Against Academic Dishonesty and the specific case of academic dishonesty according to the Hunter College Academic Integrity Procedure.

<small>I understand that all cases of academic dishonesty will be reported to the Dean of Students and all records will remain on record.</small>
Name:
English:
Email:
Signature:

# Exam Options

## Exam Times:

- Default Regular Time: Monday, 23 May, 9-11am.

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

11 December 2018

### Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commissing, attempting to commit or aiding in the commission of acts of academic dishonesty) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy Against Academic Dishonesty. Please review *Acts of Academic Dishonesty according to the Hunter College Academic Integrity Procedure*.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and all records will remain on record.
Name: _____
English: _____
Email: _____
Signature: _____

# Exam Options

## Exam Times:

- Default Regular Time: Monday, 23 May, 9-11am.
- Alternate Time: Friday, 20 May, 8am-10am.

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

14 December 2018

### Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commissing, attempting to commit or aiding in the commission of acts of academic dishonesty) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy Against Academic Dishonesty. Please review *Code of Academic Dishonesty* according to the Hunter College Academic Integrity Procedure.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and all records are permanent.
Name: _____
English: _____
Email: _____
Signature: _____

# Exam Options

## Exam Times:

- Default Regular Time: Monday, 23 May, 9-11am.
- Alternate Time: Friday, 20 May, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 29 April. If you have not done so already, email me no later than 29 April.

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

14 December 2018

### Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commissing, attempting to commit or aiding in the commission of acts of academic dishonesty) as offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy Against Academic Dishonesty and the Hunter College Code of Academic Integrity.

Indicate that all cases of academic dishonesty will be reported to the Dean of Students and all grades will be affected.
Name: _____
English: _____
Email: _____
Signature: _____

# Exam Options

## Exam Times:

- Default Regular Time: Monday, 23 May, 9-11am.
- Alternate Time: Friday, 20 May, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 29 April. If you have not done so already, email me no later than 29 April.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on 23 May.**

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

14 December 2018

### Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commissing, attempting to commit or aiding in the commission of acts of academic dishonesty) as offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy Against Academic Dishonesty and the Hunter College Code of Academic Integrity. Please see the Hunter College Academic Integrity Procedure.

<small>Understand that all cases of academic dishonesty will be reported to the Dean of Students and all grades will be affected.</small>
Name:
English:
Email:
Signature:

# Exam Options

## Exam Times:

- Default Regular Time: Monday, 23 May, 9-11am.
- Alternate Time: Friday, 20 May, 8am-10am.
- Accessibility Testing: Paperwork required. Must be completed on 29 April. If you have not done so already, email me no later than 29 April.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on 23 May.**
- If you choose to take the early date, **you will not be given access to the exam on 23 May even if you miss the early exam.**

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

14 December 2018

### Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or commissing another person to do your work) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy Against Academic Dishonesty. Please see *Code of Academic Dishonesty* according to the Hunter College Academic Integrity Procedure.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and all records will remain.
Name: _____
English: _____
Email: _____
Signature: _____

# Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

# Weekly Reminders!



Before next lecture, don't forget to:

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

# Weekly Reminders!



Before next lecture, don't forget to:

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

# Weekly Reminders!



Before next lecture, don't forget to:

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

# Weekly Reminders!



Before next lecture, don't forget to:

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

# Weekly Reminders!



Before next lecture, don't forget to:

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

# Lecture Slips & Writing Boards



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