

CSci 127: Introduction to Computer Science



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 the last day of class 10-12pm.

Frequently Asked Questions

From email and tutoring.

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

The official final is the last day of class 10-12pm.

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 the last day of class 10-12pm.

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 the last day of class 10-12pm.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *Our TA Lola is 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 the last day of class 10-12pm.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *Our TA Lola is happy to review concepts and old exam questions.*
- ▶ *There will be opportunity for some practice and an ungraded mock exam available on Gradescope.*

Today's Topics



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

Today's Topics



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

Predict what the code will do:

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

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

Python Tutor

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

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

(Demo with pythonTutor)

Design Pattern: Linear Search

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

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

- Example of **linear search**.

Design Pattern: Linear Search

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

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

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

Design Pattern: Linear Search

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

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

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

Design Pattern: Linear Search

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

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

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

Today's Topics



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

Python & Circuits Review: 9 Classes in 10 Minutes



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

Class 1: print(), loops, comments, & turtles

Class 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

Class 1: print(), loops, comments, & turtles

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

- As well as definite loops & the turtle package:

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

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

To the right of the code editor is a 'Result' window displaying a purple hexagon drawn by the turtle. The turtle has stamped a purple star at each vertex of the hexagon.

Class 1: variables, data types, more on loops & range()

Class 1: variables, data types, more on loops & range()

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

Class 1: 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

Class 1: 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

Class 1: 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

Class 1: 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

Class 1: 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 1: variables, data types, more on loops & range()

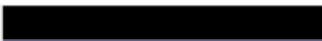
- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items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

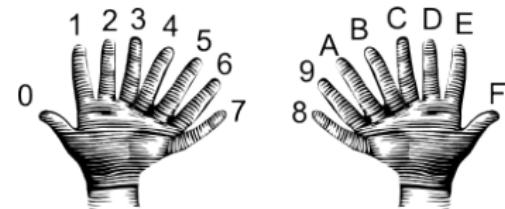
Class 1: variables, data types, more on loops & range()

- 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)
```

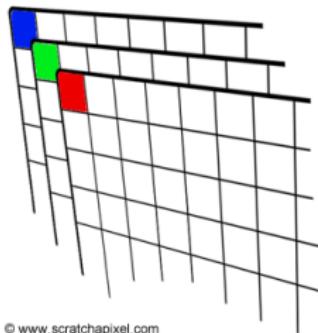
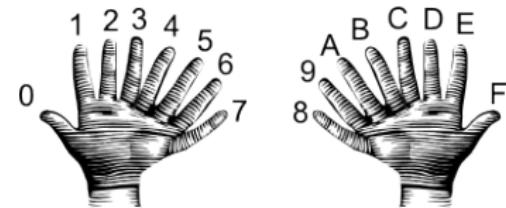
Class 2: colors, hex, slices, numpy & images

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



Class 2: colors, hex, slices, numpy & images

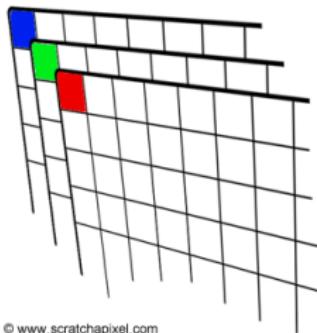
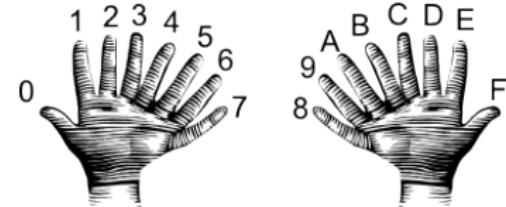
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	



© www.scratchapixel.com

Class 2: 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

Class 3: design problem (cropping images) & decisions



Class 3: design problem (cropping images) & decisions



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

Class 3: 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.

Class 3: 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.

Class 3: 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')
```

Class 4: 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.")

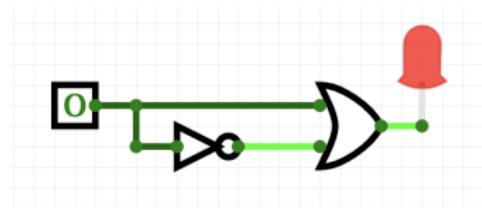
visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

Class 4: 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



Class 5: structured data, pandas, & more design

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,
All population figures are consistent with present-day boundaries.....
First census after the consolidation of the five boroughs.....

.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1690,4937,2037,,727,7881
1771,21843,36231,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,67534,6250,6844,2135,4973,85934
1820,123704,11487,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,19613,14034,5346,10965,391114
1850,35344,218912,188913,14891,5346,10965,391115
1860,613469,279122,32903,23933,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51989,33029,1911801
1890,1367111,707128,67201,58381,47444,2437204
1900,185093,116582,152999,200567,67921,2437202
1910,223342,1634351,2841,430980,8569,476683
1920,2233103,2018354,44607,44607,73201,11651,59148
1930,1867111,707128,67201,58381,47444,2437203
1940,1889924,2498285,1297634,1394711,1394441,7454995
1950,1960101,2738175,1550949,1451277,191555,7891957
1960,1696101,2319319,1890941,1451277,191555,7891984
1970,1539231,2460704,1471071,135443,7891984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2485326,2229379,1332450,419782,8080879
2010,1583873,2304705,2216722,1385108,4475153,8175133
2015,1444018,2040733,2339150,1459444,474558,8059405

nycHistPop.csv

In Lab 6

Class 5: structured data, pandas, & more design

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

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,
All population figures are consistent with present-day boundaries.....
First census after the consolidation of the five boroughs.....

.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1690,4937,2037,727,788,111
1771,21843,36231,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,67541,6211,6811,2000,4973,85934
1820,123704,11187,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,18013,14041,5348,10965,391114
1850,355441,218913,18891,5815,11581,501115
1860,813469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51980,33091,1911801
1890,1364473,71041,6348,56125,35254,2186134
1900,185093,116582,152999,200567,67621,2437202
1910,233142,1634351,2841,430980,8569,476683
1920,2210103,2018354,44601,44701,73201,11651,50148
1930,1667111,1579128,1579128,1579128,1579128,4930446
1940,1889924,2498285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1690101,2319319,1890911,1890911,1890911,781984
1970,1539231,2460711,2472171,2472171,235443,768462
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1302789,737977,7322564
2000,1537195,2485326,2229379,1332450,419728,8080879
2010,1583873,2504705,2277722,1385108,8175133
2015,1444018,2646733,2339150,1459444,474558,8059405

nycHistPop.csv

In Lab 6

Class 5: structured data, pandas, & more design

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,2017,...,727,7181,10000  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75935  
1810,68000,62000,68000,18000,45000,100000  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14000,5346,10965,391114  
1850,355400,218000,185000,58000,100000,500000  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,599403,56537,51980,33091,1911801  
1890,1380000,720000,680000,530000,310000,2000000  
1900,1850993,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22101103,2018354,44601,44601,72021,11651,500000  
1930,18671103,1796128,1796128,1796128,1796128,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690000,2000000,1800000,1600000,1400000,7000000  
1970,1539231,1860701,1747101,1747101,1745443,7000000  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,737977,7322564  
2000,1537195,2485326,2223379,1332450,419728,8080879  
2010,1583873,2504705,2216722,1385108,8175133,8175133  
2015,1444018,2646733,2339150,1459446,474558,8056405
```

nycHistPop.csv

In Lab 6

Class 5: 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.....  
Five census after the consolidation of the five boroughs.....  
.....  
Year,Population  
1698,203,2037,...,727,7181  
1771,21843,36241,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,60000,60000,60000,59374  
1820,123704,11487,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31210,10113,14000,5346,10965,391114  
1850,35549,12800,12800,12800,12800,15  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59940,5653,51980,33091,191180  
1890,1370000,1370000,1370000,1370000,1370000,134  
1900,1850593,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22421103,2018354,44600,72021,11651,50048  
1930,26671128,26671128,26671128,26671128,26671128,4930446  
1940,1889924,2498285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1696000,1696000,1696000,1696000,1696000,781984  
1970,1539231,1465701,14721473,14721701,135443,769464  
1980,1428285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,737997,77322564  
2000,1537195,2485326,2229379,1332650,419782,8080879  
2010,1583873,2504705,2277722,1385108,6175133,8175133  
2015,1444518,2436733,2339150,1459446,474558,8959405
```

nycHistPop.csv

In Lab 6

Class 5: structured data, pandas, & more design

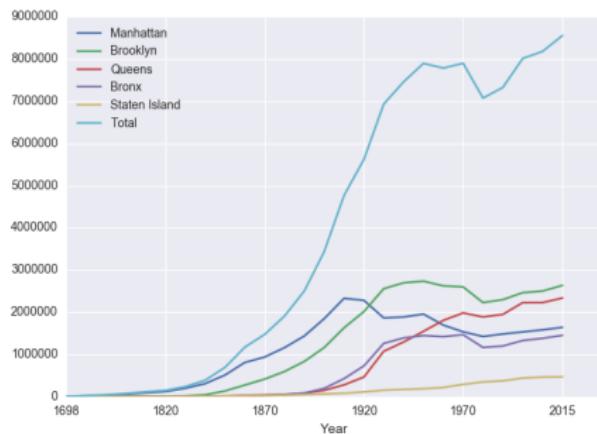
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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  
1699,Manhattan, Brooklyn, Queens, Bronx, Staten Island, Total  
1771,21843,36231,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75935  
1810,67531,5544,6241,1755,4543,75934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14041,5348,10965,391114  
1850,355441,21800,18851,5835,11515,415134  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33051,1911801  
1890,1375711,71011,6301,58100,33051,1911804  
1900,1850993,1165852,152999,200567,67921,2437202  
1910,233142,1634351,2841,430980,8569,476683  
1920,2210103,2018354,44601,44601,73201,11651,50048  
1930,2667128,2303936,1891325,1891325,1583446,4930446  
1940,188924,2690285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7991957  
1960,1696101,2303935,1890974,1890974,1581984,781984  
1970,1539231,2465701,1472701,1472701,135443,798460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,378977,7322564  
2000,1537195,2485326,2229379,1332650,413728,8080879  
2010,1583873,2504705,2216722,1385108,413728,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6



Class 6: 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()
```

Class 6: 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.

Class 6: 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()`

Class 6: 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:

Class 6: 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")`

Class 6: 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,

Class 6: 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.

Class 7: 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)
```

Class 7: 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).

Class 7: 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**.

Class 7: 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**

Class 7: 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.

Class 7: 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.

Class 8: 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)
```

Class 9: 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)
```

Class 9: 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)
```

Class 9: 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.

Class 9: 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.`

Class 9: 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: 9 Classes in 10 Minutes



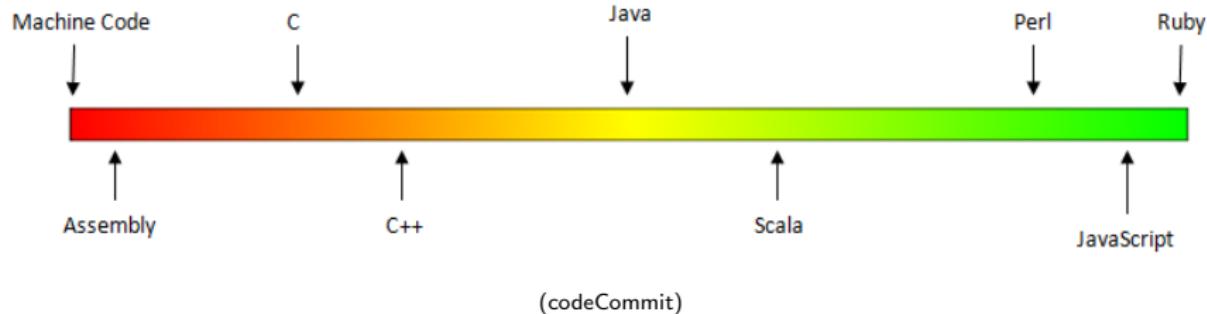
- Input/Output (I/O): `input()` and `print()`; pandas for CSV files
- Types:
 - ▶ Primitive: `int`, `float`, `bool`, `string`;
 - ▶ Container: lists (but not dictionaries/hashes or tuples)
- Objects: turtles (used but did not design our own)
- Loops: definite & indefinite
- Conditionals: `if-elif-else`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

Today's Topics



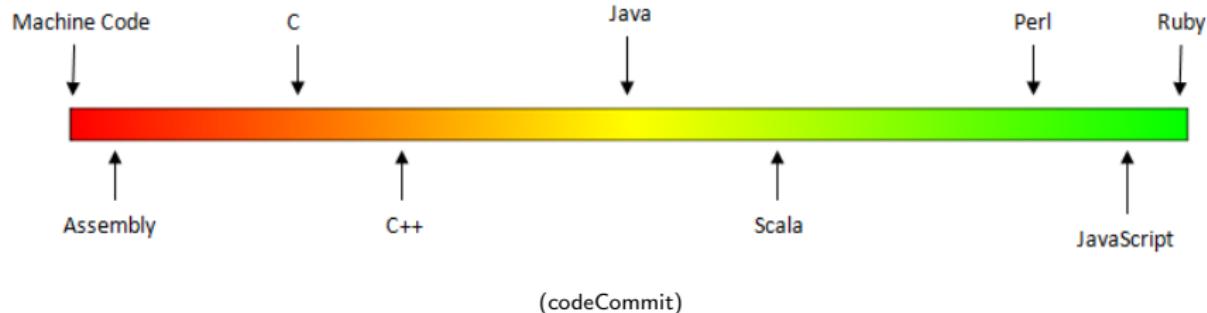
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

Low-Level vs. High-Level Languages



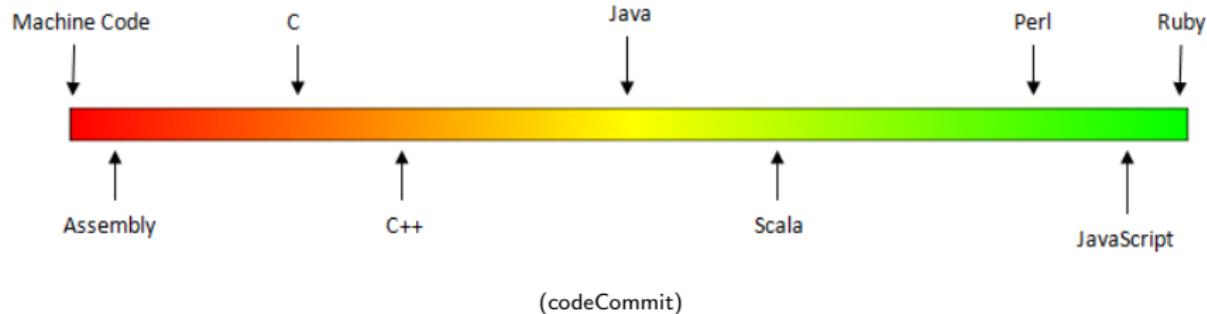
- Can view programming languages on a continuum.

Low-Level vs. High-Level Languages



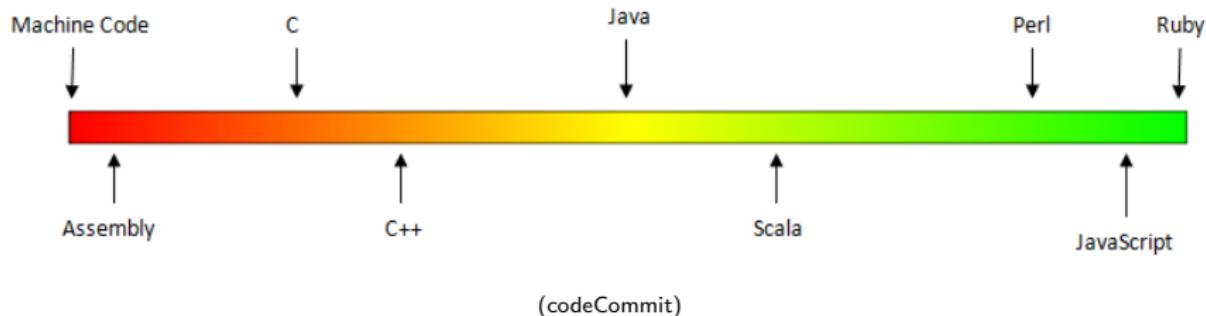
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

Low-Level vs. High-Level Languages



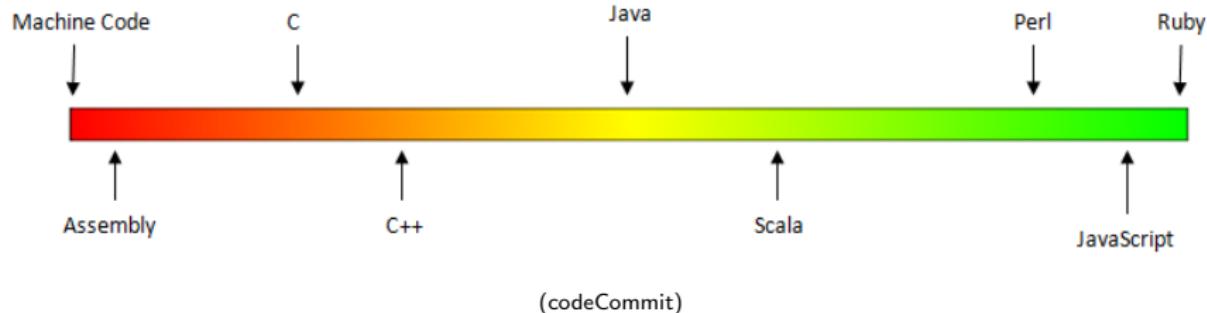
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

Low-Level vs. High-Level Languages



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

Low-Level vs. High-Level Languages

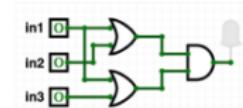


- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine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

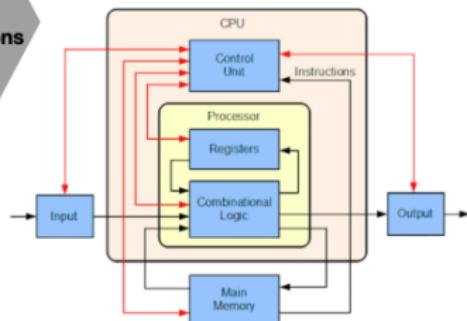
Processing

१०८ अनुवाद
प्राचीन विजयनगर के शासकों का नामांकन करने के लिए विभिन्न विधियाँ उपलब्ध हैं। इनमें से एक विधि यह है कि शासक का नाम उसकी राजधानी के नाम के साथ लिया जाए। इसके अनुसार विजयनगर के शासकों का नामांकन निम्नलिखित है—

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.

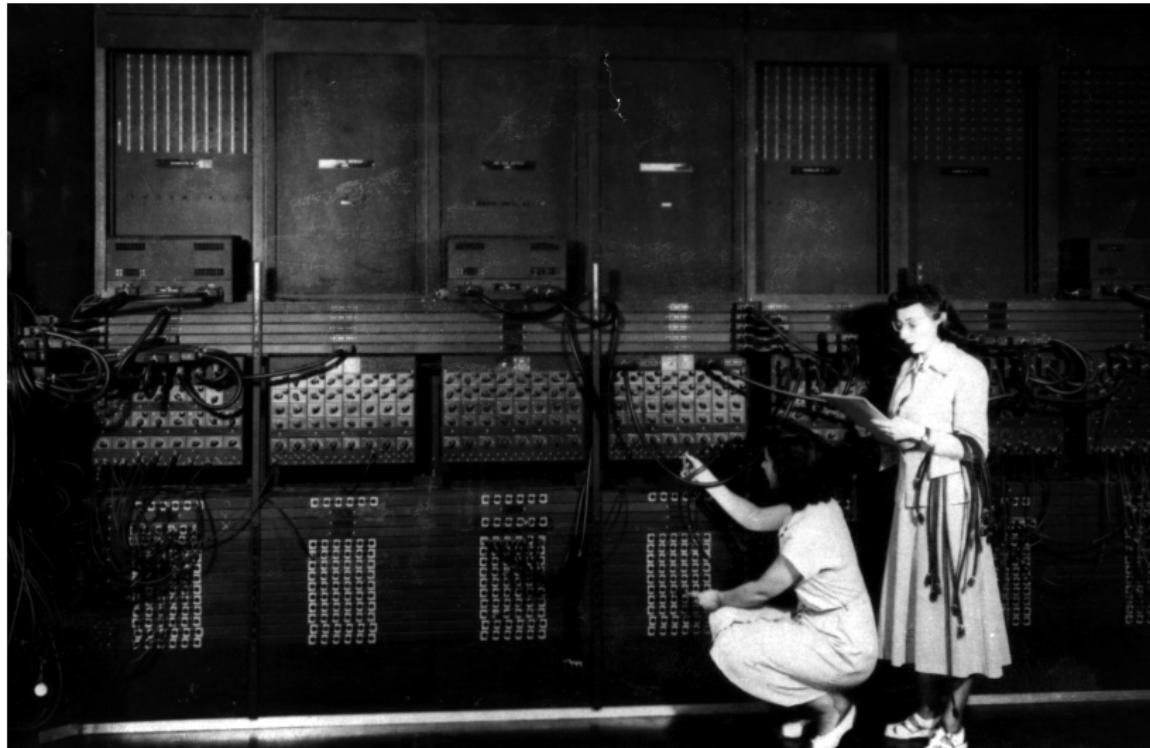


Circuits (switches)
On/Off 1/0 Logic
Billions of switches/bits



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

Machine Language



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

Machine Language

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

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

BREAK

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

(wiki)

Machine Language

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

```

A 002000 C2 3B REP #3B
A 002002 7F CLC
A 002003 FB SED
A 002004 34 32 #1234
A 002007 69 21 43 ADC #43
A 002008 69 03 7F STA #7F
A 00200E D9 CLD
A 00200F E2 3B SEP #3B
A 002011 90 BHK
A 002012

```

F

```

PB PC Mm0:012C A X Y SP DP IB
; 00 2013 00100000 0000 0000 0002 CFFF 0000 00
$ 2000

BREAK

PB PC Mm0:012C A X Y SP DP IB
; 00 2013 00100000 0550 0000 0002 CFFF 0000 00
$ 7193 7193

Mm0:012C 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

(wiki)

Machine Language

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

```

F
PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

BREAK

PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

PB PC Mm00012C A X Y SP DP IB
; 00 2013 00110800 5555 0000 0002 CFFF 0000 00
$ 2380

```

(wiki)

Machine Language

(wiki)

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

Machine Language



A screenshot of a terminal window displaying assembly code and processor registers. The assembly code is:

```
    .C2 30 REP #30
    .B 000002 FB CLD
    .B 000003 SEI
    .B 000004 69 34 12 LDa #1234
    .B 000007 69 21 43 LDc #4321
    .B 000008 8F 03 7F 01 STA #17F03
    .B 00000E 8F 03 7F 00 CLD
    .B 00000F 69 30 SEI #30
    .B 000111 80 PCK
    .B 2012

    PB PC MU#00000000 A X Y SP DP R0
    : 00000000 00000000 00000000 00000000 00000000 CFFF 00000000
    & 20000000

BREAK

    PB PC MU#00000000 A X Y SP DP R0
    : 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    & 20000000
```

The registers shown are:

Register	Value
PB	00000000
PC	00000000
MU	00000000
A	00000000
X	00000000
Y	00000000
SP	00000000
DP	00000000
R0	00000000
CLOCK	20000000

(wiki)

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

"Hello World!" in Simplified Machine Language

Line: 3 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

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

(WeMIPS)

WeMIPS

```
# Store "Hello world!" at the top of the stack
ADDI $t0, $zero, 72 # $t0
LB $t1, @t0
ADDI $t2, $zero, 101 # $t2
SB $t1, 1($t2)
LB $t3, @t2
ADDI $t4, $zero, 108 # $t4
SB $t3, 2($t4)
LB $t5, @t4
ADDI $t6, $zero, 115 # $t6
SB $t5, 3($t6)
LB $t7, @t6
ADDI $t8, $zero, 111 # $t8
SB $t7, 4($t8)
LB $t9, @t8
ADDI $t10, $zero, 32 # $(space)
SB $t9, 5($t10)
LB $t11, @t10
ADDI $t12, $zero, 113 # $t12
SB $t11, 6($t12)
LB $t13, @t12
ADDI $t14, $zero, 110 # $t14
SB $t13, 7($t14)
LB $t15, @t14
ADDI $t16, $zero, 114 # $t16
SB $t15, 8($t16)
LB $t17, @t16
ADDI $t18, $zero, 108 # $t18
SB $t17, 9($t18)
LB $t19, @t18
ADDI $t20, $zero, 109 # $t20
SB $t19, 10($t20)
LB $t21, @t20
ADDI $t22, $zero, 32 # $t22
SB $t21, 11($t22)
LB $t23, @t22
ADDI $t24, $zero, 0 # (null)
SB $t23, 12($t24)

ADDI $t0, $zero, 4 # $t0 = 4 for print string
LB $t1, @t0
SYSCALL # print to the log
```

Step	Rule	Enable auto switching			
S	T	A	V	Stack	Log
s2:			10		
s3:			9		
s2:			9		
s3:			22		
s4:			999		
s2:			976		
s3:			927		
s7:			418		

(Demo with WeMIPS)

MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there are tabs for "Show/Hide Demo", "User Guide", "Unit Tests", and "Docs". Below the tabs are navigation buttons: "Addition Counter", "Btav", "Looper", "Stack Test", "Hello World", "Code Gen Save String", "Interactive", "Binary/Decimal", "Decimal/Binary", and "Debug". The "Debug" tab is selected.

The main area displays the assembly code:

```
1 # Shows "Hello world" at the top of the stack
2 .data
3 msg: .asciiz "Hello world\n"
4 .text
5 addi $t0, $zero, 100 # $t0 = 100
6 addi $t1, $zero, 101 # $t1 = 101
7 addi $t2, $zero, 102 # $t2 = 102
8 addi $t3, $zero, 103 # $t3 = 103
9 addi $t4, $zero, 104 # $t4 = 104
10 addi $t5, $zero, 105 # $t5 = 105
11 addi $t6, $zero, 106 # $t6 = 106
12 addi $t7, $zero, 107 # $t7 = 107
13 addi $t8, $zero, 108 # $t8 = 108
14 addi $t9, $zero, 109 # $t9 = 109
15 addi $t10, $zero, 110 # $t10 = 110
16 addi $t11, $zero, 111 # $t11 = 111
17 addi $t12, $zero, 112 # $t12 = 112
18 addi $t13, $zero, 113 # $t13 = 113
19 addi $t14, $zero, 114 # $t14 = 114
20 addi $t15, $zero, 115 # $t15 = 115
21 addi $t16, $zero, 116 # $t16 = 116
22 addi $t17, $zero, 117 # $t17 = 117
23 addi $t18, $zero, 118 # $t18 = 118
24 addi $t19, $zero, 119 # $t19 = 119
25 addi $t20, $zero, 120 # $t20 = 120
26 addi $t21, $zero, 121 # $t21 = 121
27 addi $t22, $zero, 122 # $t22 = 122
28 addi $t23, $zero, 123 # $t23 = 123
29 addi $t24, $zero, 124 # $t24 = 124
30 addi $t25, $zero, 125 # $t25 = 125
31 addi $t26, $zero, 126 # $t26 = 126
32 addi $t27, $zero, 127 # $t27 = 127
33 addi $t28, $zero, 128 # $t28 = 128
34 addi $t29, $zero, 129 # $t29 = 129
35 addi $t30, $zero, 130 # $t30 = 130
36 addi $t31, $zero, 131 # $t31 = 131
37 addi $t32, $zero, 132 # $t32 = 132
38 addi $t33, $zero, 133 # $t33 = 133
39 addi $t34, $zero, 134 # $t34 = 134
40 addi $t35, $zero, 135 # $t35 = 135
41 addi $t36, $zero, 136 # $t36 = 136
42 addi $t37, $zero, 137 # $t37 = 137
43 addi $t38, $zero, 138 # $t38 = 138
44 addi $t39, $zero, 139 # $t39 = 139
45 addi $t40, $zero, 140 # $t40 = 140
46 addi $t41, $zero, 141 # $t41 = 141
47 addi $t42, $zero, 142 # $t42 = 142
48 addi $t43, $zero, 143 # $t43 = 143
49 addi $t44, $zero, 144 # $t44 = 144
50 addi $t45, $zero, 145 # $t45 = 145
51 addi $t46, $zero, 146 # $t46 = 146
52 addi $t47, $zero, 147 # $t47 = 147
53 addi $t48, $zero, 148 # $t48 = 148
54 addi $t49, $zero, 149 # $t49 = 149
55 addi $t50, $zero, 150 # $t50 = 150
56 addi $t51, $zero, 151 # $t51 = 151
57 addi $t52, $zero, 152 # $t52 = 152
58 addi $t53, $zero, 153 # $t53 = 153
59 addi $t54, $zero, 154 # $t54 = 154
60 addi $t55, $zero, 155 # $t55 = 155
61 addi $t56, $zero, 156 # $t56 = 156
62 addi $t57, $zero, 157 # $t57 = 157
63 addi $t58, $zero, 158 # $t58 = 158
64 addi $t59, $zero, 159 # $t59 = 159
65 addi $t60, $zero, 160 # $t60 = 160
66 addi $t61, $zero, 161 # $t61 = 161
67 addi $t62, $zero, 162 # $t62 = 162
68 addi $t63, $zero, 163 # $t63 = 163
69 addi $t64, $zero, 164 # $t64 = 164
70 addi $t65, $zero, 165 # $t65 = 165
71 addi $t66, $zero, 166 # $t66 = 166
72 addi $t67, $zero, 167 # $t67 = 167
73 addi $t68, $zero, 168 # $t68 = 168
74 addi $t69, $zero, 169 # $t69 = 169
75 addi $t70, $zero, 170 # $t70 = 170
76 addi $t71, $zero, 171 # $t71 = 171
77 addi $t72, $zero, 172 # $t72 = 172
78 addi $t73, $zero, 173 # $t73 = 173
79 addi $t74, $zero, 174 # $t74 = 174
80 addi $t75, $zero, 175 # $t75 = 175
81 addi $t76, $zero, 176 # $t76 = 176
82 addi $t77, $zero, 177 # $t77 = 177
83 addi $t78, $zero, 178 # $t78 = 178
84 addi $t79, $zero, 179 # $t79 = 179
85 addi $t80, $zero, 180 # $t80 = 180
86 addi $t81, $zero, 181 # $t81 = 181
87 addi $t82, $zero, 182 # $t82 = 182
88 addi $t83, $zero, 183 # $t83 = 183
89 addi $t84, $zero, 184 # $t84 = 184
90 addi $t85, $zero, 185 # $t85 = 185
91 addi $t86, $zero, 186 # $t86 = 186
92 addi $t87, $zero, 187 # $t87 = 187
93 addi $t88, $zero, 188 # $t88 = 188
94 addi $t89, $zero, 189 # $t89 = 189
95 addi $t90, $zero, 190 # $t90 = 190
96 addi $t91, $zero, 191 # $t91 = 191
97 addi $t92, $zero, 192 # $t92 = 192
98 addi $t93, $zero, 193 # $t93 = 193
99 addi $t94, $zero, 194 # $t94 = 194
100 addi $t95, $zero, 195 # $t95 = 195
101 addi $t96, $zero, 196 # $t96 = 196
102 addi $t97, $zero, 197 # $t97 = 197
103 addi $t98, $zero, 198 # $t98 = 198
104 addi $t99, $zero, 199 # $t99 = 199
105 addi $t100, $zero, 200 # $t100 = 200
106 addi $t101, $zero, 201 # $t101 = 201
107 addi $t102, $zero, 202 # $t102 = 202
108 addi $t103, $zero, 203 # $t103 = 203
109 addi $t104, $zero, 204 # $t104 = 204
110 addi $t105, $zero, 205 # $t105 = 205
111 addi $t106, $zero, 206 # $t106 = 206
112 addi $t107, $zero, 207 # $t107 = 207
113 addi $t108, $zero, 208 # $t108 = 208
114 addi $t109, $zero, 209 # $t109 = 209
115 addi $t110, $zero, 210 # $t110 = 210
116 addi $t111, $zero, 211 # $t111 = 211
117 addi $t112, $zero, 212 # $t112 = 212
118 addi $t113, $zero, 213 # $t113 = 213
119 addi $t114, $zero, 214 # $t114 = 214
120 addi $t115, $zero, 215 # $t115 = 215
121 addi $t116, $zero, 216 # $t116 = 216
122 addi $t117, $zero, 217 # $t117 = 217
123 addi $t118, $zero, 218 # $t118 = 218
124 addi $t119, $zero, 219 # $t119 = 219
125 addi $t120, $zero, 220 # $t120 = 220
126 addi $t121, $zero, 221 # $t121 = 221
127 addi $t122, $zero, 222 # $t122 = 222
128 addi $t123, $zero, 223 # $t123 = 223
129 addi $t124, $zero, 224 # $t124 = 224
130 addi $t125, $zero, 225 # $t125 = 225
131 addi $t126, $zero, 226 # $t126 = 226
132 addi $t127, $zero, 227 # $t127 = 227
133 addi $t128, $zero, 228 # $t128 = 228
134 addi $t129, $zero, 229 # $t129 = 229
135 addi $t130, $zero, 230 # $t130 = 230
136 addi $t131, $zero, 231 # $t131 = 231
137 addi $t132, $zero, 232 # $t132 = 232
138 addi $t133, $zero, 233 # $t133 = 233
139 addi $t134, $zero, 234 # $t134 = 234
140 addi $t135, $zero, 235 # $t135 = 235
141 addi $t136, $zero, 236 # $t136 = 236
142 addi $t137, $zero, 237 # $t137 = 237
143 addi $t138, $zero, 238 # $t138 = 238
144 addi $t139, $zero, 239 # $t139 = 239
145 addi $t140, $zero, 240 # $t140 = 240
146 addi $t141, $zero, 241 # $t141 = 241
147 addi $t142, $zero, 242 # $t142 = 242
148 addi $t143, $zero, 243 # $t143 = 243
149 addi $t144, $zero, 244 # $t144 = 244
150 addi $t145, $zero, 245 # $t145 = 245
151 addi $t146, $zero, 246 # $t146 = 246
152 addi $t147, $zero, 247 # $t147 = 247
153 addi $t148, $zero, 248 # $t148 = 248
154 addi $t149, $zero, 249 # $t149 = 249
155 addi $t150, $zero, 250 # $t150 = 250
156 addi $t151, $zero, 251 # $t151 = 251
157 addi $t152, $zero, 252 # $t152 = 252
158 addi $t153, $zero, 253 # $t153 = 253
159 addi $t154, $zero, 254 # $t154 = 254
160 addi $t155, $zero, 255 # $t155 = 255
161 addi $t156, $zero, 256 # $t156 = 256
162 addi $t157, $zero, 257 # $t157 = 257
163 addi $t158, $zero, 258 # $t158 = 258
164 addi $t159, $zero, 259 # $t159 = 259
165 addi $t160, $zero, 260 # $t160 = 260
166 addi $t161, $zero, 261 # $t161 = 261
167 addi $t162, $zero, 262 # $t162 = 262
168 addi $t163, $zero, 263 # $t163 = 263
169 addi $t164, $zero, 264 # $t164 = 264
170 addi $t165, $zero, 265 # $t165 = 265
171 addi $t166, $zero, 266 # $t166 = 266
172 addi $t167, $zero, 267 # $t167 = 267
173 addi $t168, $zero, 268 # $t168 = 268
174 addi $t169, $zero, 269 # $t169 = 269
175 addi $t170, $zero, 270 # $t170 = 270
176 addi $t171, $zero, 271 # $t171 = 271
177 addi $t172, $zero, 272 # $t172 = 272
178 addi $t173, $zero, 273 # $t173 = 273
179 addi $t174, $zero, 274 # $t174 = 274
180 addi $t175, $zero, 275 # $t175 = 275
181 addi $t176, $zero, 276 # $t176 = 276
182 addi $t177, $zero, 277 # $t177 = 277
183 addi $t178, $zero, 278 # $t178 = 278
184 addi $t179, $zero, 279 # $t179 = 279
185 addi $t180, $zero, 280 # $t180 = 280
186 addi $t181, $zero, 281 # $t181 = 281
187 addi $t182, $zero, 282 # $t182 = 282
188 addi $t183, $zero, 283 # $t183 = 283
189 addi $t184, $zero, 284 # $t184 = 284
190 addi $t185, $zero, 285 # $t185 = 285
191 addi $t186, $zero, 286 # $t186 = 286
192 addi $t187, $zero, 287 # $t187 = 287
193 addi $t188, $zero, 288 # $t188 = 288
194 addi $t189, $zero, 289 # $t189 = 289
195 addi $t190, $zero, 290 # $t190 = 290
196 addi $t191, $zero, 291 # $t191 = 291
197 addi $t192, $zero, 292 # $t192 = 292
198 addi $t193, $zero, 293 # $t193 = 293
199 addi $t194, $zero, 294 # $t194 = 294
200 addi $t195, $zero, 295 # $t195 = 295
201 addi $t196, $zero, 296 # $t196 = 296
202 addi $t197, $zero, 297 # $t197 = 297
203 addi $t198, $zero, 298 # $t198 = 298
204 addi $t199, $zero, 299 # $t199 = 299
205 addi $t200, $zero, 300 # $t200 = 300
206 addi $t201, $zero, 301 # $t201 = 301
207 addi $t202, $zero, 302 # $t202 = 302
208 addi $t203, $zero, 303 # $t203 = 303
209 addi $t204, $zero, 304 # $t204 = 304
210 addi $t205, $zero, 305 # $t205 = 305
211 addi $t206, $zero, 306 # $t206 = 306
212 addi $t207, $zero, 307 # $t207 = 307
213 addi $t208, $zero, 308 # $t208 = 308
214 addi $t209, $zero, 309 # $t209 = 309
215 addi $t210, $zero, 310 # $t210 = 310
216 addi $t211, $zero, 311 # $t211 = 311
217 addi $t212, $zero, 312 # $t212 = 312
218 addi $t213, $zero, 313 # $t213 = 313
219 addi $t214, $zero, 314 # $t214 = 314
220 addi $t215, $zero, 315 # $t215 = 315
221 addi $t216, $zero, 316 # $t216 = 316
222 addi $t217, $zero, 317 # $t217 = 317
223 addi $t218, $zero, 318 # $t218 = 318
224 addi $t219, $zero, 319 # $t219 = 319
225 addi $t220, $zero, 320 # $t220 = 320
226 addi $t221, $zero, 321 # $t221 = 321
227 addi $t222, $zero, 322 # $t222 = 322
228 addi $t223, $zero, 323 # $t223 = 323
229 addi $t224, $zero, 324 # $t224 = 324
230 addi $t225, $zero, 325 # $t225 = 325
231 addi $t226, $zero, 326 # $t226 = 326
232 addi $t227, $zero, 327 # $t227 = 327
233 addi $t228, $zero, 328 # $t228 = 328
234 addi $t229, $zero, 329 # $t229 = 329
235 addi $t230, $zero, 330 # $t230 = 330
236 addi $t231, $zero, 331 # $t231 = 331
237 addi $t232, $zero, 332 # $t232 = 332
238 addi $t233, $zero, 333 # $t233 = 333
239 addi $t234, $zero, 334 # $t234 = 334
240 addi $t235, $zero, 335 # $t235 = 335
241 addi $t236, $zero, 336 # $t236 = 336
242 addi $t237, $zero, 337 # $t237 = 337
243 addi $t238, $zero, 338 # $t238 = 338
244 addi $t239, $zero, 339 # $t239 = 339
245 addi $t240, $zero, 340 # $t240 = 340
246 addi $t241, $zero, 341 # $t241 = 341
247 addi $t242, $zero, 342 # $t242 = 342
248 addi $t243, $zero, 343 # $t243 = 343
249 addi $t244, $zero, 344 # $t244 = 344
250 addi $t245, $zero, 345 # $t245 = 345
251 addi $t246, $zero, 346 # $t246 = 346
252 addi $t247, $zero, 347 # $t247 = 347
253 addi $t248, $zero, 348 # $t248 = 348
254 addi $t249, $zero, 349 # $t249 = 349
255 addi $t250, $zero, 350 # $t250 = 350
256 addi $t251, $zero, 351 # $t251 = 351
257 addi $t252, $zero, 352 # $t252 = 352
258 addi $t253, $zero, 353 # $t253 = 353
259 addi $t254, $zero, 354 # $t254 = 354
260 addi $t255, $zero, 355 # $t255 = 355
261 addi $t256, $zero, 356 # $t256 = 356
262 addi $t257, $zero, 357 # $t257 = 357
263 addi $t258, $zero, 358 # $t258 = 358
264 addi $t259, $zero, 359 # $t259 = 359
265 addi $t260, $zero, 360 # $t260 = 360
266 addi $t261, $zero, 361 # $t261 = 361
267 addi $t262, $zero, 362 # $t262 = 362
268 addi $t263, $zero, 363 # $t263 = 363
269 addi $t264, $zero, 364 # $t264 = 364
270 addi $t265, $zero, 365 # $t265 = 365
271 addi $t266, $zero, 366 # $t266 = 366
272 addi $t267, $zero, 367 # $t267 = 367
273 addi $t268, $zero, 368 # $t268 = 368
274 addi $t269, $zero, 369 # $t269 = 369
275 addi $t270, $zero, 370 # $t270 = 370
276 addi $t271, $zero, 371 # $t271 = 371
277 addi $t272, $zero, 372 # $t272 = 372
278 addi $t273, $zero, 373 # $t273 = 373
279 addi $t274, $zero, 374 # $t274 = 374
280 addi $t275, $zero, 375 # $t275 = 375
281 addi $t276, $zero, 376 # $t276 = 376
282 addi $t277, $zero, 377 # $t277 = 377
283 addi $t278, $zero, 378 # $t278 = 378
284 addi $t279, $zero, 379 # $t279 = 379
285 addi $t280, $zero, 380 # $t280 = 380
286 addi $t281, $zero, 381 # $t281 = 381
287 addi $t282, $zero, 382 # $t282 = 382
288 addi $t283, $zero, 383 # $t283 = 383
289 addi $t284, $zero, 384 # $t284 = 384
290 addi $t285, $zero, 385 # $t285 = 385
291 addi $t286, $zero, 386 # $t286 = 386
292 addi $t287, $zero, 387 # $t287 = 387
293 addi $t288, $zero, 388 # $t288 = 388
294 addi $t289, $zero, 389 # $t289 = 389
295 addi $t290, $zero, 390 # $t290 = 390
296 addi $t291, $zero, 391 # $t291 = 391
297 addi $t292, $zero, 392 # $t292 = 392
298 addi $t293, $zero, 393 # $t293 = 393
299 addi $t294, $zero, 394 # $t294 = 394
290 addi $t295, $zero, 395 # $t295 = 395
291 addi $t296, $zero, 396 # $t296 = 396
292 addi $t297, $zero, 397 # $t297 = 397
293 addi $t298, $zero, 398 # $t298 = 398
294 addi $t299, $zero, 399 # $t299 = 399
295 addi $t290, $zero, 400 # $t290 = 400
296 addi $t291, $zero, 401 # $t291 = 401
297 addi $t292, $zero, 402 # $t292 = 402
298 addi $t293, $zero, 403 # $t293 = 403
299 addi $t294, $zero, 404 # $t294 = 404
290 addi $t295, $zero, 405 # $t295 = 405
291 addi $t296, $zero, 406 # $t296 = 406
292 addi $t297, $zero, 407 # $t297 = 407
293 addi $t298, $zero, 408 # $t298 = 408
294 addi $t299, $zero, 409 # $t299 = 409
295 addi $t290, $zero, 410 # $t290 = 410
296 addi $t291, $zero, 411 # $t291 = 411
297 addi $t292, $zero, 412 # $t292 = 412
298 addi $t293, $zero, 413 # $t293 = 413
299 addi $t294, $zero, 414 # $t294 = 414
290 addi $t295, $zero, 415 # $t295 = 415
291 addi $t296, $zero, 416 # $t296 = 416
292 addi $t297, $zero, 417 # $t297 = 417
293 addi $t298, $zero, 418 # $t298 = 418
294 addi $t299, $zero, 419 # $t299 = 419
295 addi $t290, $zero, 420 # $t290 = 420
296 addi $t291, $zero, 421 # $t291 = 421
297 addi $t292, $zero, 422 # $t292 = 422
298 addi $t293, $zero, 423 # $t293 = 423
299 addi $t294, $zero, 424 # $t294 = 424
290 addi $t295, $zero, 425 # $t295 = 425
291 addi $t296, $zero, 426 # $t296 = 426
292 addi $t297, $zero, 427 # $t297 = 427
293 addi $t298, $zero, 428 # $t298 = 428
294 addi $t299, $zero, 429 # $t299 = 429
295 addi $t290, $zero, 430 # $t290 = 430
296 addi $t291, $zero, 431 # $t291 = 431
297 addi $t292, $zero, 432 # $t292 = 432
298 addi $t293, $zero, 433 # $t293 = 433
299 addi $t294, $zero, 434 # $t294 = 434
290 addi $t295, $zero, 435 # $t295 = 435
291 addi $t296, $zero, 436 # $t296 = 436
292 addi $t297, $zero, 437 # $t297 = 437
293 addi $t298, $zero, 438 # $t298 = 438
294 addi $t299, $zero, 439 # $t299 = 439
295 addi $t290, $zero, 440 # $t290 = 440
296 addi $t291, $zero, 441 # $t291 = 441
297 addi $t292, $zero, 442 # $t292 = 442
298 addi $t293, $zero, 443 # $t293 = 443
299 addi $t294, $zero, 444 # $t294 = 444
290 addi $t295, $zero, 445 # $t295 = 445
291 addi $t296, $zero, 446 # $t296 = 446
292 addi $t297, $zero, 447 # $t297 = 447
293 addi $t298, $zero, 448 # $t298 = 448
294 addi $t299, $zero, 449 # $t299 = 449
295 addi $t290, $zero, 450 # $t290 = 450
296 addi $t291, $zero, 451 # $t291 = 451
297 addi $t292, $zero, 452 # $t292 = 452
298 addi $t293, $zero, 453 # $t293 = 453
299 addi $t294, $zero, 454 # $t294 = 454
290 addi $t295, $zero, 455 # $t295 = 455
291 addi $t296, $zero, 456 # $t296 = 456
292 addi $t297, $zero, 457 # $t297 = 457
293 addi $t298, $zero, 458 # $t298 = 458
294 addi $t299, $zero, 459 # $t299 = 459
295 addi $t290, $zero, 460 # $t290 = 460
296 addi $t291, $zero, 461 # $t291 = 461
297 addi $t292, $zero, 462 # $t292 = 462
298 addi $t293, $zero, 463 # $t293 = 463
299 addi $t294, $zero, 464 # $t294 = 464
290 addi $t295, $zero, 465 # $t295 = 465
291 addi $t296, $zero, 466 # $t296 = 466
292 addi $t297, $zero, 467 # $t297 = 467
293 addi $t298, $zero, 468 # $t298 = 468
294 addi $t299, $zero, 469 # $t299 = 469
295 addi $t290, $zero, 470 # $t290 = 470
296 addi $t291, $zero, 471 # $t291 = 471
297 addi $t292, $zero, 472 # $t292 = 472
298 addi $t293, $zero, 473 # $t293 = 473
299 addi $t294, $zero, 474 # $t294 = 474
290 addi $t295, $zero, 475 # $t295 = 475
291 addi $t296, $zero, 476 # $t296 = 476
292 addi $t297, $zero, 477 # $t297 = 477
293 addi $t298, $zero, 478 # $t298 = 478
294 addi $t299, $zero, 479 # $t299 = 479
295 addi $t290, $zero, 480 # $t290 = 480
296 addi $t291, $zero, 481 # $t291 = 481
297 addi $t292, $zero, 482 # $t292 = 482
298 addi $t293, $zero, 483 # $t293 = 483
299 addi $t294, $zero, 484 # $t294 = 484
290 addi $t295, $zero, 485 # $t295 = 485
291 addi $t296, $zero, 486 # $t296 = 486
292 addi $t297, $zero, 487 # $t297 = 487
293 addi $t298, $zero, 488 # $t298 = 488
294 addi $t299, $zero, 489 # $t299 = 489
295 addi $t290, $zero, 490 # $t290 = 490
296 addi $t291, $zero, 491 # $t291 = 491
297 addi $t292, $zero, 492 # $t292 = 492
298 addi $t293, $zero, 493 # $t293 = 493
299 addi $t294, $zero, 494 # $t294 = 494
290 addi $t295, $zero, 495 # $t295 = 495
291 addi $t296, $zero, 496 # $t296 = 496
292 addi $t297, $zero, 497 # $t297 = 497
293 addi $t298, $zero, 498 # $t298 = 498
294 addi $t299, $zero, 499 # $t299 = 499
295 addi $t290, $zero, 500 # $t290 = 500
296 addi $t291, $zero, 501 # $t291 = 501
297 addi $t292, $zero, 502 # $t292 = 502
298 addi $t293, $zero, 503 # $t293 = 503
299 addi $t294, $zero, 504 # $t294 = 504
290 addi $t295, $zero, 505 # $t295 = 505
291 addi $t296, $zero, 506 # $t296 = 506
292 addi $t297, $zero, 507 # $t297 = 507
293 addi $t298, $zero, 508 # $t298 = 508
294 addi $t299, $zero, 509 # $t299 = 509
295 addi $t290, $zero, 510 # $t290 = 510
296 addi $t291, $zero, 511 # $t291 = 511
297 addi $t292, $zero, 512 # $t292 = 512
298 addi $t293, $zero, 513 # $t293 = 513
299 addi $t294, $zero, 514 # $t294 = 514
290 addi $t295, $zero, 515 # $t295 = 
```

MIPS Commands

The screenshot shows a window titled "StackFrame Demo". At the top, there are tabs for "User", "3", "64", "ShowFrame Demo", "Addition Counter", "Itiva", "Looper", "Stack Test", "Hello World", "Code Gen Save String", "Interactive", "Binary Decimal", "Decimal Binary", and "Debug". Below the tabs is a text area containing MIPS assembly code:

```
1 # Shows "Hello world!" at the top of the stack
2 .text
3 .globl _start
4 _start:
5    li $t0, 0x484f4d4c # 'Hello'
6    li $t1, 0x6f6f6f6f # 11111111
7    add $t2, $t0, $t1 # t2 = Hello + 11111111
8    add $t3, $t2, $t1 # t3 = Hello + 22222222
9    add $t4, $t3, $t1 # t4 = Hello + 33333333
10   add $t5, $t4, $t1 # t5 = Hello + 44444444
11   add $t6, $t5, $t1 # t6 = Hello + 55555555
12   add $t7, $t6, $t1 # t7 = Hello + 66666666
13   add $t8, $t7, $t1 # t8 = Hello + 77777777
14   sb $t0, $t1($sp) # store 'Hello' at the top of the stack
15   li $t9, 0x00000000 # 0
16   sb $t9, 4($sp) # print a new line
17   li $t10, 0x4e454545 # 'syscall'
18   sb $t10, 8($sp) # print to the log
19   sb $t10, 12($sp) # print to the log
20   sb $t10, 16($sp) # print to the log
21   sb $t10, 20($sp) # print to the log
22   sb $t10, 24($sp) # print to the log
23   sb $t10, 28($sp) # print to the log
24   sb $t10, 32($sp) # print to the log
25   sb $t10, 36($sp) # print to the log
26   sb $t10, 40($sp) # print to the log
27   sb $t10, 44($sp) # print to the log
28   sb $t10, 48($sp) # print to the log
29   sb $t10, 52($sp) # print to the log
30   sb $t10, 56($sp) # print to the log
31   sb $t10, 60($sp) # print to the log
32   syscall
```

To the right of the assembly code is a table titled "Registers" with columns S, T, A, V, Stack, and Log. The table shows the following register values:

	S	T	A	V	Stack	Log
\$0					10	
\$1					9	
\$2					8	
\$3					7	
\$4					6	
\$5					5	
\$6					4	
\$7					3	
\$8					2	
\$9					1	
\$10					0	
\$11					418	
\$12					417	

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

MIPS Commands

The screenshot shows a software interface for assembly language development. At the top, there's a menu bar with 'File', 'Edit', 'Run', 'Help', 'ShowFide Demo', 'Addition Counter', 'Stack Test', 'Looper', 'Stack Test -', 'Hello World', 'Code Gen Save String', 'Interactive', 'Binary Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with icons for 'Run', 'Break', 'Stop', 'Step', 'Reset', and 'Run' (with a 'c' icon). To the right of the toolbar are links for 'User Guide', 'Unit Tests', and 'Docs'. The main area contains assembly code and a register dump.

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

MIPS Commands

The screenshot shows the StackFrame Demo application interface. At the top, there's a menu bar with 'File', 'Edit', 'Run', 'Help', 'User Guide', 'Unit Tests', and 'Doc'. Below the menu is a toolbar with buttons for 'Addition', 'Calculator', 'Itav', 'Looper', 'Stack Test', 'Hello World', 'Code Gen Save String', 'Interactive', 'Binary Decimal', 'Decimal Binary', and 'Debug'. The main area contains two panes: one for assembly code and one for register values.

Assembly Code:

```
1 # Shows "Hello world!" at the top of the stack
2 .text
3 .globl _start
4 _start:
5    addiu   $sp,$sp,-16      # allocate space for arguments
6    addiu   $sp,$sp,-16      # allocate space for local variables
7    addiu   $sp,$sp,-16      # allocate space for local variables
8    addiu   $sp,$sp,-16      # allocate space for local variables
9    addiu   $sp,$sp,-16      # allocate space for local variables
10   addiu   $sp,$sp,-16      # allocate space for local variables
11   addiu   $sp,$sp,-16      # allocate space for local variables
12   addiu   $sp,$sp,-16      # allocate space for local variables
13   addiu   $sp,$sp,-16      # allocate space for local variables
14   addiu   $sp,$sp,-16      # allocate space for local variables
15   addiu   $sp,$sp,-16      # allocate space for local variables
16   addiu   $sp,$sp,-16      # allocate space for local variables
17   addiu   $sp,$sp,-16      # allocate space for local variables
18   addiu   $sp,$sp,-16      # allocate space for local variables
19   addiu   $sp,$sp,-16      # allocate space for local variables
20   addiu   $sp,$sp,-16      # allocate space for local variables
21   addiu   $sp,$sp,-16      # allocate space for local variables
22   addiu   $sp,$sp,-16      # allocate space for local variables
23   addiu   $sp,$sp,-16      # allocate space for local variables
24   addiu   $sp,$sp,-16      # allocate space for local variables
25   addiu   $sp,$sp,-16      # allocate space for local variables
26   addiu   $sp,$sp,-16      # allocate space for local variables
27   addiu   $sp,$sp,-16      # allocate space for local variables
28   addiu   $sp,$sp,-16      # allocate space for local variables
29   addiu   $sp,$sp,-16      # allocate space for local variables
30   addiu   $sp,$sp,-16      # allocate space for local variables
31   addiu   $sp,$sp,-16      # allocate space for local variables
32   addiu   $sp,$sp,-16      # allocate space for local variables
33   addiu   $sp,$sp,-16      # print to the log
34   syscall
```

Registers:

S	T	A	V	Stack	Log
s0				10	
s1				9	
s2				8	
s3				7	
s4				6	
s5				5	
s6				4	
s7				3	
t0				2	
t1				1	
t2				0	
t3				418	

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

MIPS Commands

The screenshot shows the StackFrame Demo interface. At the top, there are tabs for User, Show, and Go, along with links for ShowFrame Demo, Addition, Decoder, Itav, Looper, Stack Test, Hello World, Code Gen, Save String, Interactive, Binary, Decimal, Debug, and Hexadecimal. Below these are tabs for Stack, Decimal, and Binary.

The main area displays assembly code:

```
1 # Shows "Hello world" at the top of the stack
2 .text
3 .globl _start
4 _start:
5    addi   $s0, $zero, 111 # a
6    addi   $s1, $zero, 110 # b
7    addi   $t0, $zero, 100 # 1
8    addi   $t1, $zero, 100 # 1
9    addi   $t2, $zero, 100 # 1
10   addi   $t3, $zero, 100 # 1
11   addi   $t4, $zero, 100 # 1
12   addi   $t5, $zero, 100 # 1
13   addi   $t6, $zero, 100 # 1
14   addi   $t7, $zero, 100 # 1
15   addi   $t8, $zero, 100 # 1
16   addi   $t9, $zero, 100 # 1
17   addi   $t10, $zero, 111 # n
18   addi   $t11, $zero, 110 # v
19   addi   $t12, $zero, 100 # l
20   addi   $t13, $zero, 100 # o
21   addi   $t14, $zero, 100 # l
22   addi   $t15, $zero, 100 # w
23   addi   $t16, $zero, 100 # r
24   addi   $t17, $zero, 100 # l
25   addi   $t18, $zero, 100 # d
26   addi   $t19, $zero, 100 # e
27   addi   $t20, $zero, 0 # (null)
28   addi   $t21, $zero, 0 # (null)
29   addi   $t22, $zero, 0 # (null)
30   addi   $t23, $zero, 0 # (null)
31   addi   $t24, $zero, 0 # (null)
32   addi   $t25, $zero, 0 # print to the log
33   syscall
```

To the right, there is a stack dump table:

S	T	A	V	Stack	Log
s0				11	
s1				9	
s2				8	
s3				22	
s4				60	
s5				61	
s6				807	
s7				418	

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

MIPS Commands

The screenshot shows the ShowMIPS Demo application window. At the top, there are tabs for User, ShowMIPS Demo, and Run. Below the tabs are buttons for Addition, Counter, IfElse, Looper, StackTest, Hello World, CodeGen, SaveString, Interactive, Binary, Decimal, and Debug. The main area contains assembly code and a register dump.

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

S	T	A	V	Stack	Log
\$0				10	
\$1				9	
\$2				8	
\$3				7	
\$4				6	
\$5				5	
\$6				4	
\$7				3	
\$8				2	
\$9				1	
\$10				0	
\$11				418	
\$12					
\$13					
\$14					
\$15					
\$16					
\$17					
\$18					
\$19					
\$20					
\$21					
\$22					
\$23					
\$24					
\$25					
\$26					
\$27					
\$28					
\$29					
\$30					
\$31					

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

MIPS Commands

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

MIPS Commands

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

MIPS Commands

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

Challenge:

Line: 3 Go! Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

Step Run Enable auto switching

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

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

WeMIPS

(Demo with WeMIPS)

Today's Topics



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

Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



A screenshot of a hex editor application. The left pane shows a list of memory pages, each containing a series of memory addresses and their corresponding byte values. The right pane is a detailed view of a specific page, showing memory addresses from 0x00000000 to 0x0000000F. The bytes displayed are: 48 45 4C 4C 4D 4E 4F 4F 4A 4B 4C 4D 4E 4F 4F 4A 4B 4C. Below the address column, there is a column labeled 'Label' which contains the labels 'start' and 'loop'. The bottom of the window has a status bar with the text 'File Edit View Insert Tools Options Help New Page, Set Hex View'.

Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Jump Demo

Line: 18 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

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

(Demo
with
WeMIPS)

Step Run Enable auto switching

S T A V Stack Log

Clear Log

Emulation complete, returning to line 1

abcdefghijklmnopqrstuvwxyz

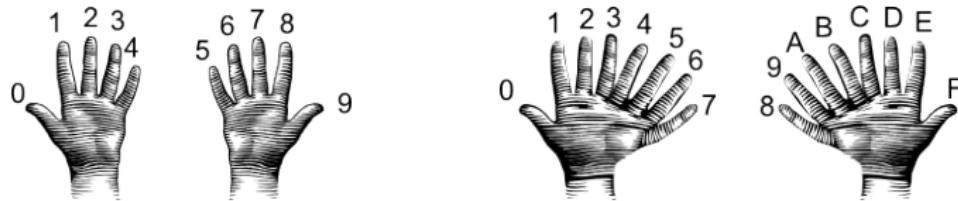


Today's Topics



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

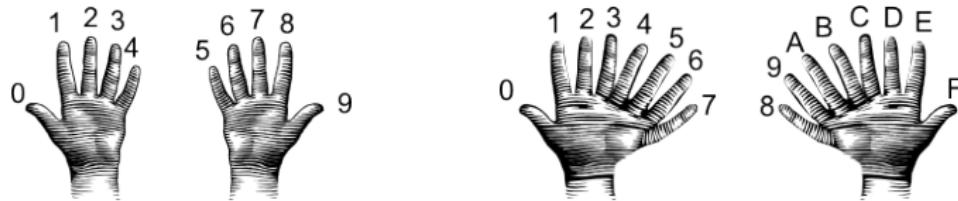
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

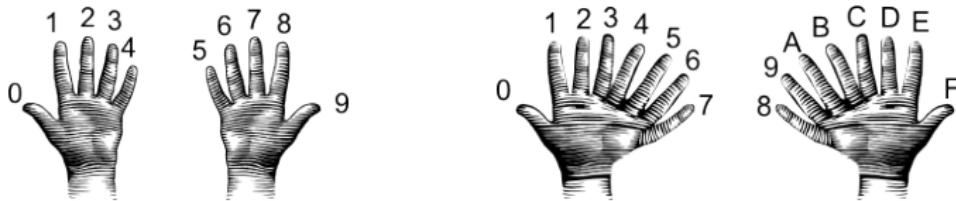
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

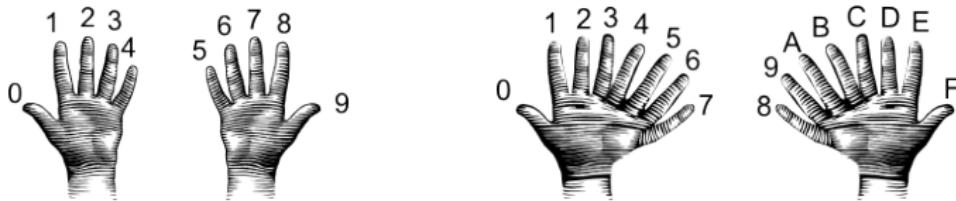
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

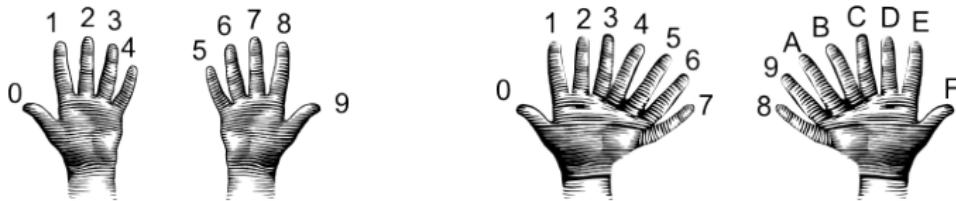
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

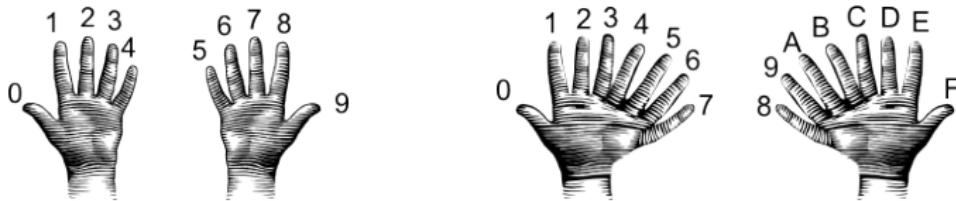
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

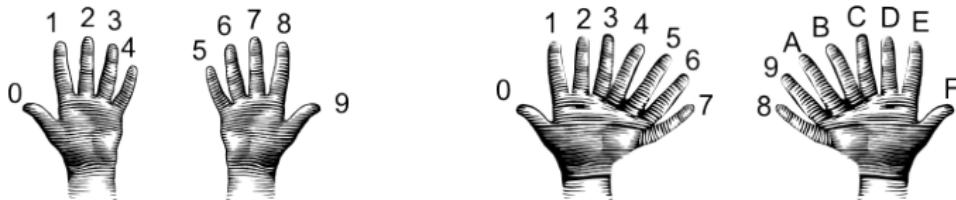
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

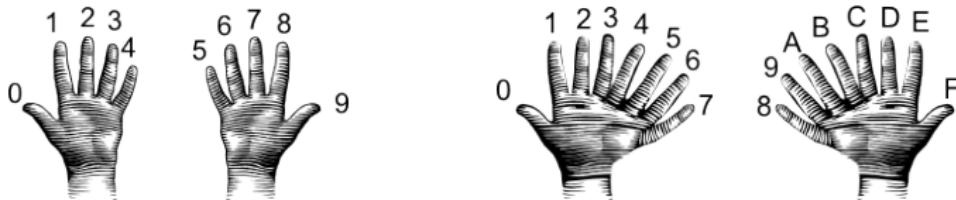
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

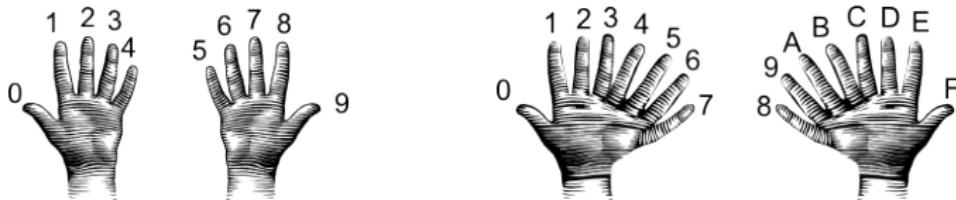
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

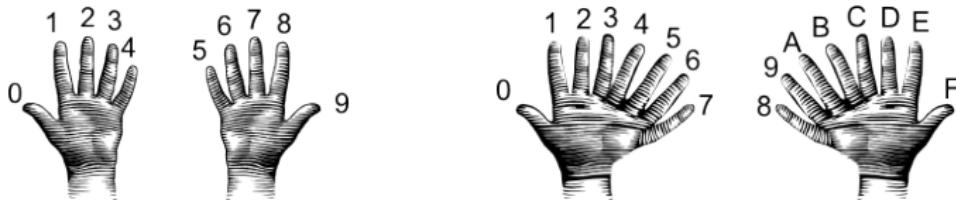
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

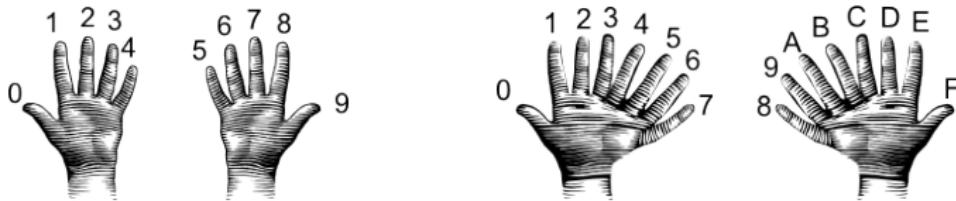
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

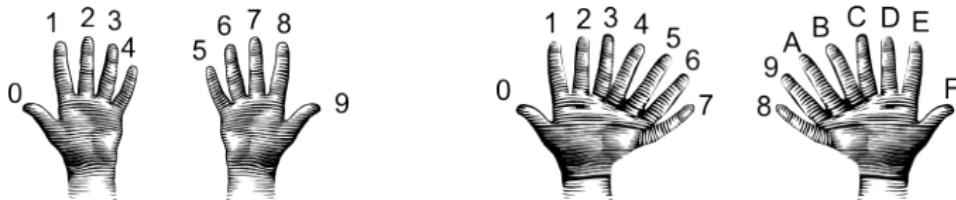
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

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

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

Answer is 42.

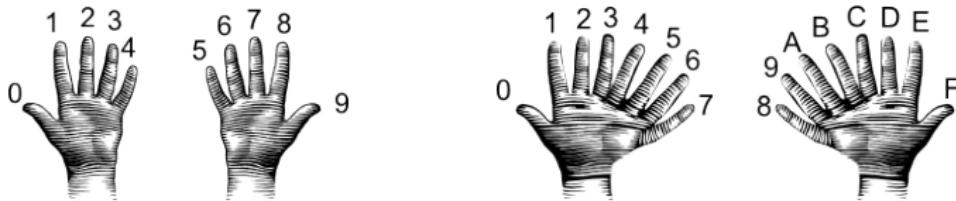
- Example: what is 99 as a decimal number?

9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

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

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

Answer is 42.

- Example: what is 99 as a decimal number?

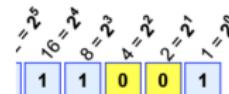
9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Answer is 153.

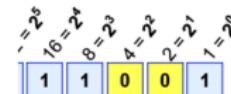
Decimal to Binary: Converting Between Bases



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

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

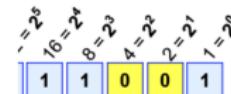
Decimal to Binary: Converting Between Bases



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

- 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.

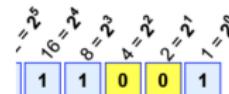
Decimal to Binary: Converting Between Bases



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.

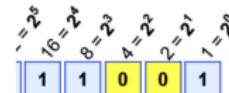
Decimal to Binary: Converting Between Bases



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

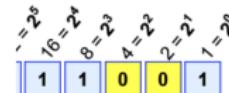


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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

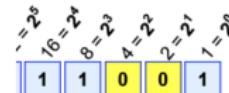


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

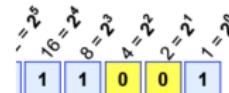


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

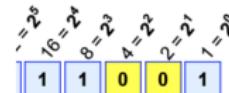


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

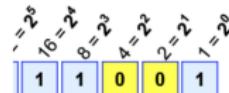


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases



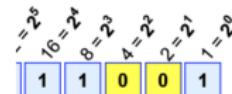
- From decimal to binary:

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

- Example: what is 130 in binary notation?

130/128 is 1 rem 2.

Decimal to Binary: Converting Between Bases



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

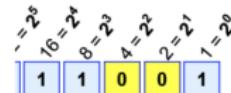
- From decimal to binary:

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

- Example: what is 130 in binary notation?

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

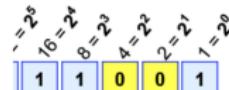
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

- Example: what is 130 in binary notation?

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

2/64 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

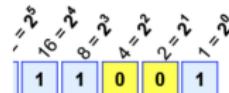
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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



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

- From decimal to binary:

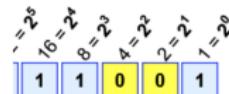
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

- Example: what is 130 in binary notation?

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

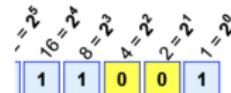
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

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

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

2/32 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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.

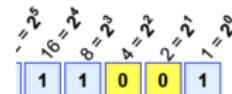
- Example: what is 130 in binary notation?

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:

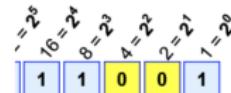
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next 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: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- 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?

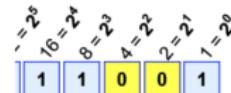
130/128 is 1 rem 2. First digit is 1: 1...

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?

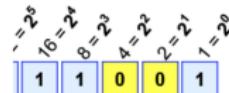
130/128 is 1 rem 2. First digit is 1: 1...

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



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

- From decimal to binary:

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

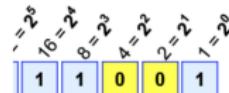
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

Decimal to Binary: Converting Between Bases



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

- 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...

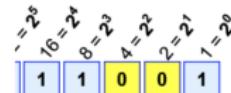
2/64 is 0 rem 2. Next digit is 0: 10...

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...

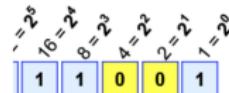
2/64 is 0 rem 2. Next digit is 0: 10...

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

- From decimal to binary:

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

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

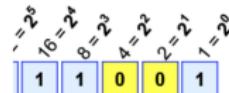
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

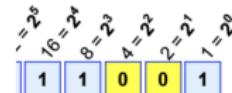
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

2/4 is 0 remainder 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

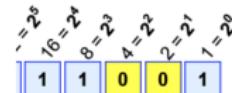
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

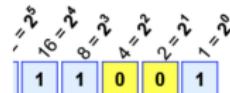
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

Decimal to Binary: Converting Between Bases



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

- 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...

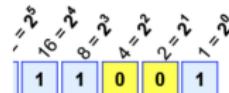
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

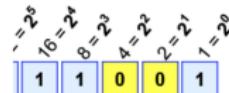
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

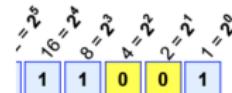
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

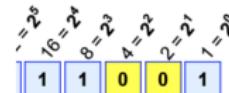
2/8 is 0 rem 2. Next digit is 0: 10000...

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...

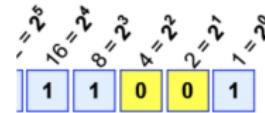
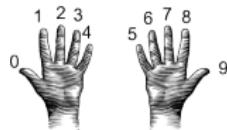
2/8 is 0 rem 2. Next digit is 0: 10000...

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

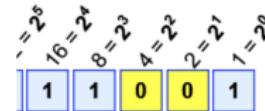
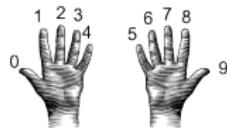
Decimal to Binary: Converting Between Bases



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

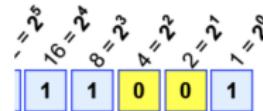
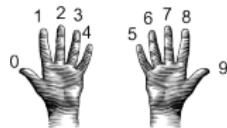


Example: $1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99/128$ is 0 rem 99.

Decimal to Binary: Converting Between Bases

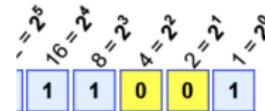
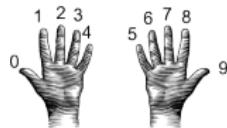


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases



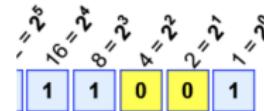
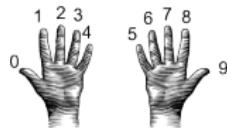
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

99/64 is 1 rem 35.

Decimal to Binary: Converting Between Bases



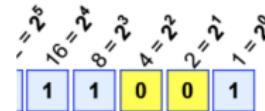
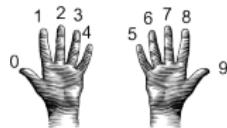
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

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

Decimal to Binary: Converting Between Bases



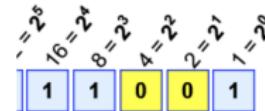
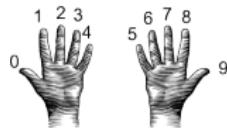
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

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

Decimal to Binary: Converting Between Bases



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

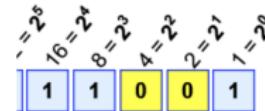
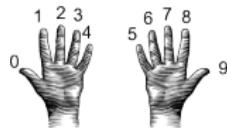
- Example: what is 99 in binary notation?

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

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

35/32 is 1 rem 3.

Decimal to Binary: Converting Between Bases



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

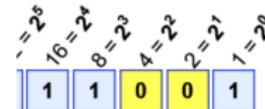
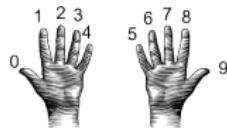
- Example: what is 99 in binary notation?

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

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

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

Decimal to Binary: Converting Between Bases



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

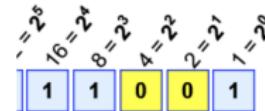
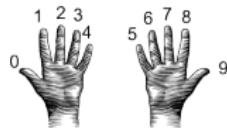
- Example: what is 99 in binary notation?

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

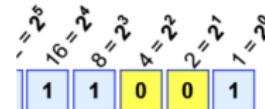
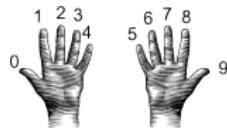
99/128 is 0 rem 99. First digit is 0: 0...

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

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

3/16 is 0 rem 3.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

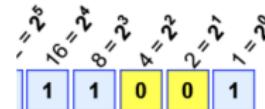
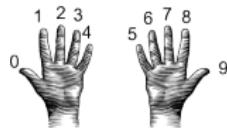
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

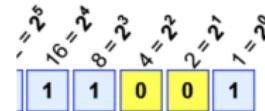
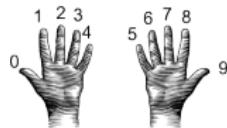
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

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...

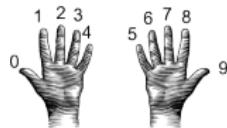
99/64 is 1 rem 35. Next digit is 1: 01...

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

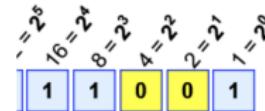
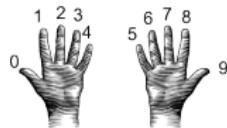
3/16 is 0 rem 3. Next digit is 0: 0110...

3/8 is 0 rem 3.

Decimal to Binary: Converting Between Bases



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...

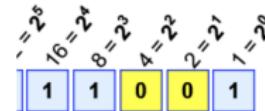
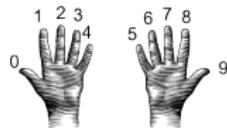
99/64 is 1 rem 35. Next digit is 1: 01...

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...

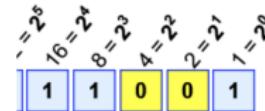
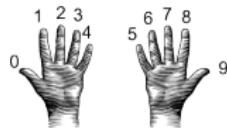
35/32 is 1 rem 3. Next digit is 1: 011...

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

- Example: what is 99 in binary notation?

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

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

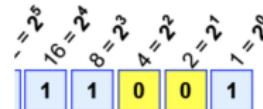
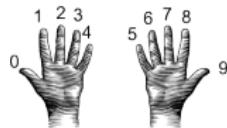
35/32 is 1 rem 3. Next digit is 1: 011...

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...

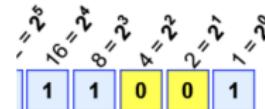
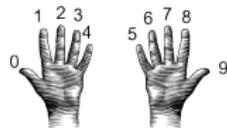
35/32 is 1 rem 3. Next digit is 1: 011...

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

- Example: what is 99 in binary notation?

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

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

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

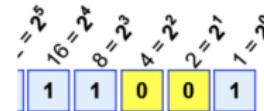
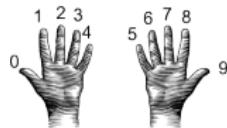
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

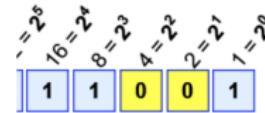
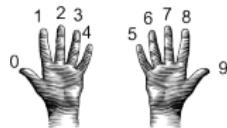
3/16 is 0 rem 3. Next digit is 0: 0110...

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

- Example: what is 99 in binary notation?

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

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

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

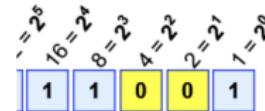
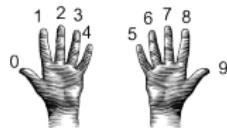
3/16 is 0 rem 3. Next digit is 0: 0110...

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...

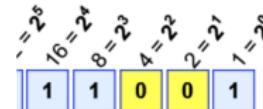
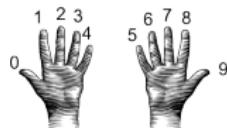
3/8 is 0 rem 3. Next digit is 0: 01100...

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

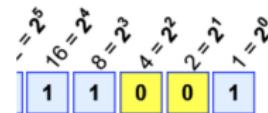
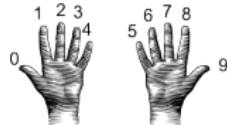


- Example: what is 99 in binary notation?

99/128 is 0 rem 99. First digit is 0:	0...
99/64 is 1 rem 35. Next digit is 1:	01...
35/32 is 1 rem 3. Next digit is 1:	011...
3/16 is 0 rem 3. Next digit is 0:	0110...
3/8 is 0 rem 3. Next digit is 0:	01100...
3/4 is 0 remainder 3. Next digit is 0:	011000...
3/2 is 1 rem 1. Next digit is 1:	0110001...
Adding the last remainder:	01100011

Answer is 1100011.

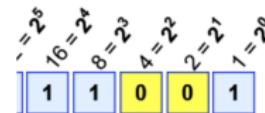
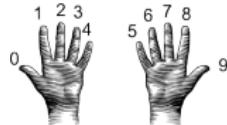
Binary to Decimal: Converting Between Bases



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

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

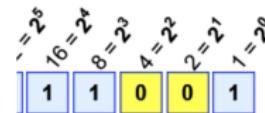
Binary to Decimal: Converting Between Bases



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

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

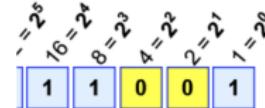
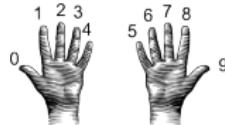
Binary to Decimal: Converting Between Bases



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

- 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

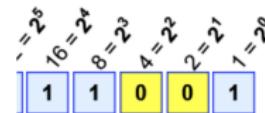
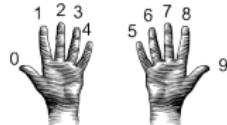


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

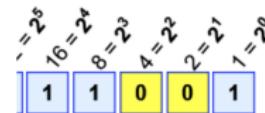
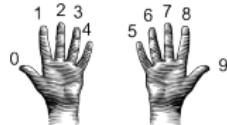


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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.

Binary to Decimal: Converting Between Bases

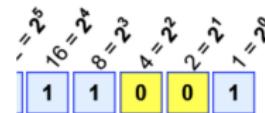


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

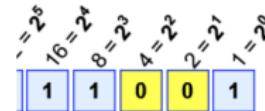
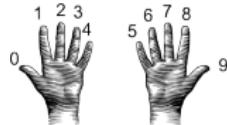


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

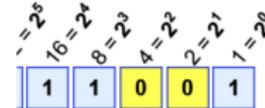


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

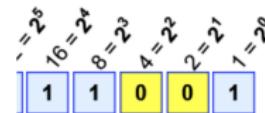
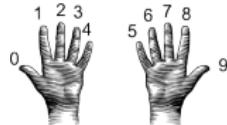


Example: $1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Binary to Decimal: Converting Between Bases



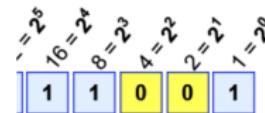
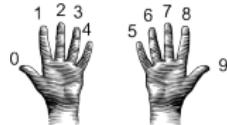
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

Sum starts with:

Binary to Decimal: Converting Between Bases



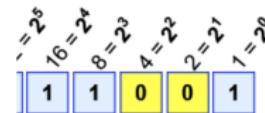
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



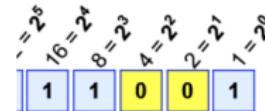
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

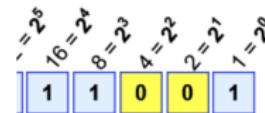
- Set sum = last digit.
- Multiply next digit by 2^1 . Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$. Add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

Sum starts with: 1

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

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

Binary to Decimal: Converting Between Bases



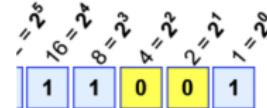
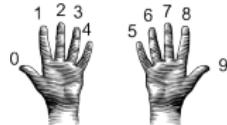
$$\text{Example: } 1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

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

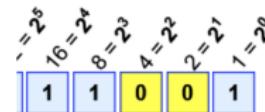
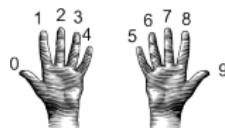
Sum starts with: 1

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

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

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

Binary to Decimal: Converting Between Bases



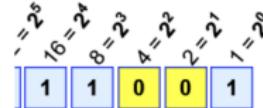
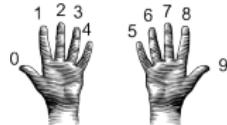
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 = 16 + 8 + 4 + 0 + 0 = 28$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



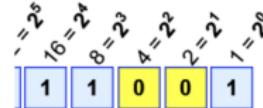
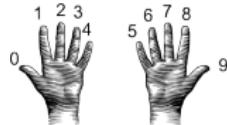
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



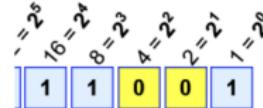
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



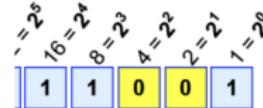
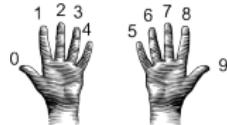
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



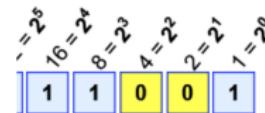
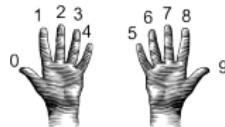
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



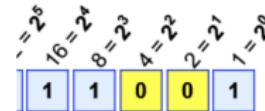
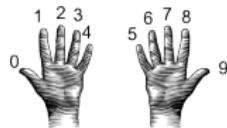
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases

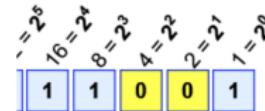
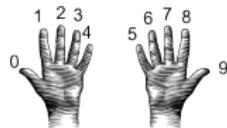


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

Binary to Decimal: Converting Between Bases



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

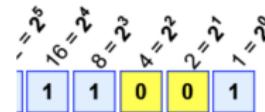
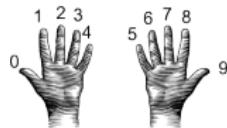
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

Binary to Decimal: Converting Between Bases



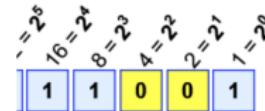
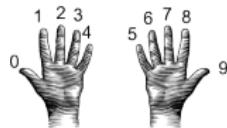
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

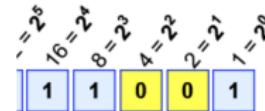
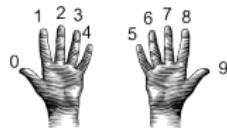
- Example: What is 10100100 in decimal?

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

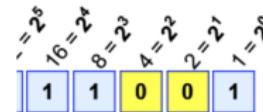
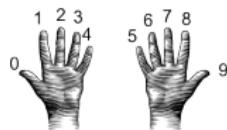
- Example: What is 10100100 in decimal?

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?

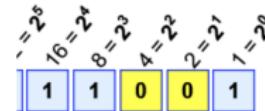
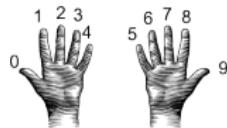
Sum starts with: 0

$0 \times 2 = 0$. 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

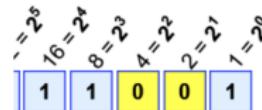
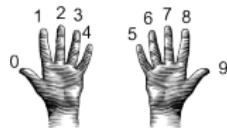


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

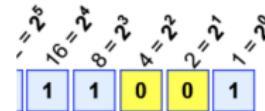
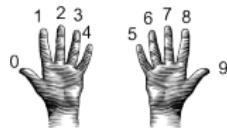
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

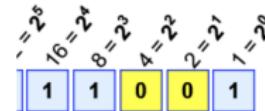
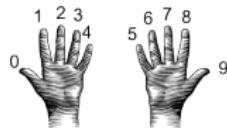
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



- Example: What is 10100100 in decimal?

Sum starts with: 0

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

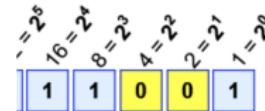
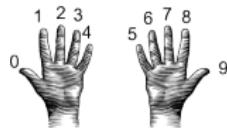
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases

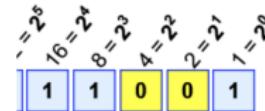
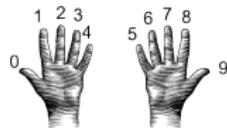


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

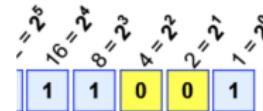
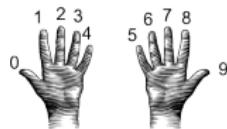
$0 \times 8 = 0$. Add 0 to sum: 4

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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

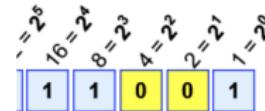
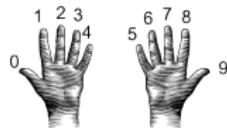
$0 \times 8 = 0$. Add 0 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

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

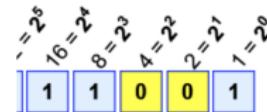
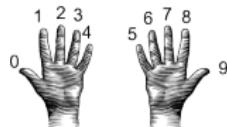
$0 \times 16 = 0$. Add 0 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases

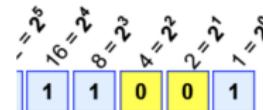
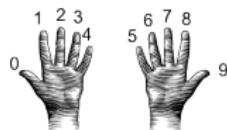


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases



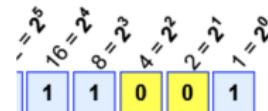
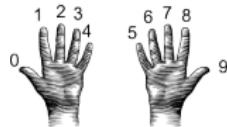
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

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

The answer is 164.

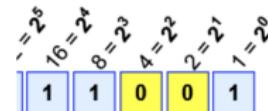
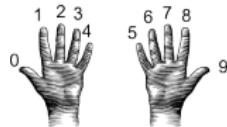
Design Challenge: Incrementers



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

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

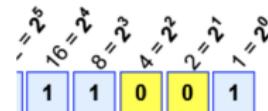
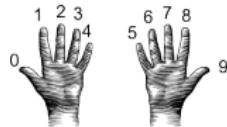
Design Challenge: Incrementers



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

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

Design Challenge: Incrementers

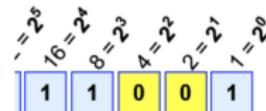
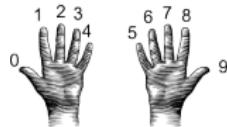


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

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

Design Challenge: Incrementers



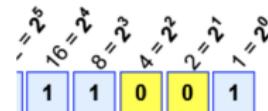
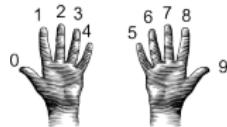
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



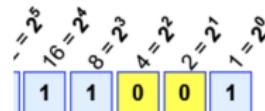
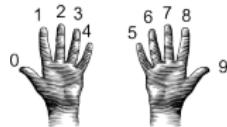
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

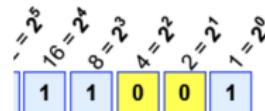
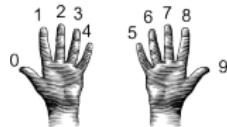
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



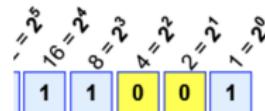
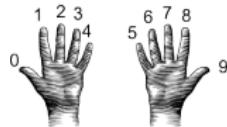
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Today's Topics



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

Final Overview: Administration

- The exam will be administered through Gradescope.

Final Overview: Administration

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

Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only during the time of the exam
- The exam format:
 - ▶ Like a long Lab Quiz, you scroll down to answer all questions.

Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only on during the time of the exam
- The exam format:
 - ▶ Like a long Lab Quiz, you scroll down to answer all questions.
 - ▶ Questions roughly correspond to the 10 parts from old exams, but will appear as a larger number of questions on Gradescope

Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only on during the time of the exam
- The exam format:
 - ▶ Like a long Lab Quiz, you scroll down to answer all questions.
 - ▶ Questions roughly correspond to the 10 parts from old exams, but will appear as a larger number of questions on Gradescope
 - ▶ Questions are variations on the programming assignments, lab exercises, and lecture design challenges.

Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only on during the time of the exam
- The exam format:
 - ▶ Like a long Lab Quiz, you scroll down to answer all questions.
 - ▶ Questions roughly correspond to the 10 parts from old exams, but will appear as a larger number of questions on Gradescope
 - ▶ Questions are variations on the programming assignments, lab exercises, and lecture design challenges.

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Class Reminders!



Before next lecture, don't forget to:

- Review this class's Lecture and Lab

Class Reminders!



Before next lecture, don't forget to:

- Review this class's Lecture and Lab
- Take the Lab Quiz on Gradescope by 9pm on Today

Class Reminders!



Before next lecture, don't forget to:

- Review this class's Lecture and Lab
- Take the Lab Quiz on Gradescope by 9pm on Today
- Submit this class's programming assignments (programs 50-52)

Class Reminders!



Before next lecture, don't forget to:

- Review this class's Lecture and Lab
- Take the Lab Quiz on Gradescope by 9pm on Today
- Submit this class's programming assignments (programs 50-52)
- At any point, visit our TA for help!!!