

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

- This lecture will be recorded

Announcements

- Thanksgiving Break starts in 9 days.



Announcements

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



Announcements



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

Announcements



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

Frequently Asked Questions

From email and tutoring.

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

Frequently Asked Questions

From email and tutoring.

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

Frequently Asked Questions

From email and tutoring.

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

Frequently Asked Questions

From email and tutoring.

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

Frequently Asked Questions

From email and tutoring.

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

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

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

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

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

Frequently Asked Questions

From email and tutoring.

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

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

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

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

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

Instead of a review sheet, we have:

Frequently Asked Questions

From email and tutoring.

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

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

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

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

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

Instead of a review sheet, we have:

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

Frequently Asked Questions

From email and tutoring.

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

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

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

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

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

Instead of a review sheet, we have:

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

Frequently Asked Questions

From email and tutoring.

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

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

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

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

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

Instead of a review sheet, we have:

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

Today's Topics



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

Today's Topics



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

Predict what the code will do:

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

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

Python Tutor

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

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

(Demo with pythonTutor)

Design Pattern: Linear Search

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

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

- Example of **linear search**.

Design Pattern: Linear Search

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

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

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

Design Pattern: Linear Search

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

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

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

Design Pattern: Linear Search

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

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

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

Today's Topics



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

Python & Circuits Review: 10 Weeks in 10 Minutes



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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

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

- Introduced comments & print():

```
#Name: Thomas Hunter
```

← These lines are comments

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

← (for us, not computer to read)

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

← (this one also)

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

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

- As well as definite loops & the turtle package:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

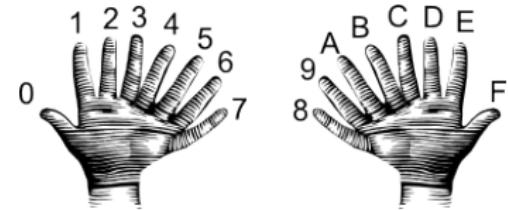
Week 2: variables, data types, more on loops & range()

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

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

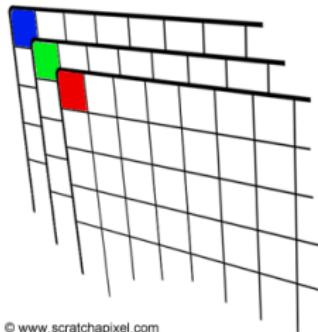
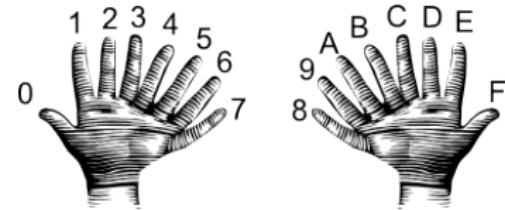
Week 3: colors, hex, slices, numpy & images

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



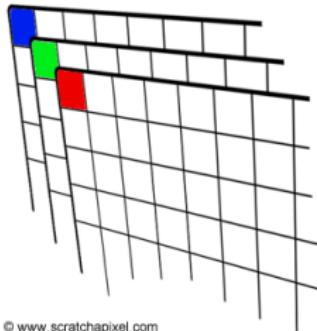
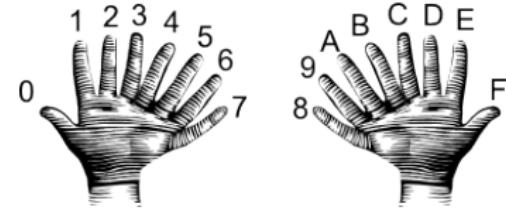
Week 3: colors, hex, slices, numpy & images

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



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

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



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

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

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

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

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

Week 4: design problem (cropping images) & decisions



Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions

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

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

Week 5: logical operators, truth tables & logical circuits

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

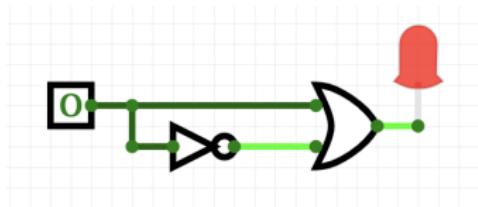
visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

Week 5: logical operators, truth tables & logical circuits

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

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

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



Week 6: structured data, pandas, & more design

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Bronx,Brooklyn,Queens,Bronx,Staten Island,Totals  
1890,1937,2017,727,788,1000  
1871,21843,26232,,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75915  
1810,61541,5854,6542,1755,4543,75934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14031,5348,10965,391114  
1850,355411,21801,18931,5815,11581,44115  
1860,513469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1367111,70000,65000,57000,33091,2148014  
1900,1850593,116582,152999,200567,67621,2437202  
1910,2233142,1634351,264041,430980,8569,4766803  
1920,2211103,2018354,446031,446031,73201,1165150,58148  
1930,1867128,2203936,1797128,1797128,15821,4930446  
1940,1889924,2499285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7891957  
1960,1696101,2319319,1899049,1451277,191555,7891984  
1970,1539231,2465701,1471701,135443,7891984  
1980,1426285,2230936,1891325,1168972,252121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419782,8080879  
2010,1583873,2504705,2271722,1385108,447512,8175133  
2015,1444518,2636733,2339150,1459444,474558,8059405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,
All population figures are consistent with present-day boundaries.....
Five census after the consolidation of the five boroughs.....
.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1890,4937,2037,,727,7881,28423
1870,33131,4549,6159,1781,3827,49447
1860,60515,5740,6442,1755,4543,75955
1850,55545,5254,5851,1715,4293,74934
1820,123704,11187,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,19113,14081,5348,10965,391114
1850,35545,21890,18591,5348,10965,401115
1860,813469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59945,5653,51980,33029,1911801
1890,1364473,66451,5653,51980,33029,205354
1900,185093,116582,152999,200567,67921,2437202
1910,2233142,1634351,284041,430980,8569,4766803
1920,2210110,2018354,446071,446071,73201,11651,50048
1930,1867137,1867137,1867137,1867137,1867137,4930446
1940,1889924,2469285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1690101,1690101,1690101,1690101,1690101,781984
1970,1359231,1359231,1359231,1359231,1359231,781984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,379977,7322564
2000,1537195,2485326,2223379,1332450,419728,8080879
2010,1583873,2504705,2272722,1385108,4175133,8175133
2015,1444018,2436733,2339150,1459444,474558,8059405

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Borough,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,2017,...,727,7181  
1771,21843,36232,...,2847,28423  
1790,33131,4549,...,6159,1781,3827,49447  
1800,60515,5740,...,6442,1755,4543,75955  
1810,67500,6200,...,6442,1755,4543,75934  
1820,123704,11187,...,8246,2792,6135,152056  
1830,20589,20535,...,9049,3023,7082,242278  
1840,31100,10113,...,14000,5346,10965,391114  
1850,35549,...,128900,...,14856,10965,391114  
1860,613469,...,279122,...,23993,...,25492,174777  
1870,942292,...,419921,...,45468,...,37393,...,33029,...,1479103  
1880,1164473,...,59943,...,5653,...,51980,...,39301,...,1911801  
1890,1364473,...,70000,...,65000,...,51980,...,39301,...,1911804  
1900,1850593,...,116582,...,152999,...,200567,...,67921,...,2437202  
1910,2233142,...,1634351,...,2841,...,430980,...,8569,...,476683  
1920,22101103,...,2018354,...,446000,...,732013,...,116582,...,591083  
1930,26671103,...,2018354,...,446000,...,732013,...,116582,...,591046  
1940,...,1889924,...,2690285,...,1297634,...,1394711,...,174441,...,7454995  
1950,...,1960101,...,2738175,...,1550949,...,1451277,...,191555,...,7991957  
1960,...,1660101,...,2738175,...,1550949,...,1451277,...,191555,...,7991984  
1970,...,1660101,...,2738175,...,1550949,...,1451277,...,191555,...,7991984  
1980,...,1426285,...,2230936,...,1891325,...,1168972,...,352121,...,7071639  
1990,...,1487536,...,2300664,...,1951598,...,1302789,...,379977,...,7322564  
2000,...,1537195,...,2485326,...,2229379,...,1332650,...,419782,...,8080879  
2010,...,1583873,...,2504705,...,2272722,...,1385108,...,474558,...,8175133  
2015,...,1444018,...,2540733,...,2339150,...,1459444,...,474558,...,8159405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Population  
1690,203,2037,...,727,7181  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,71031,6354,7041,1755,4543,75934  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,311510,11013,14041,5348,10965,391114  
1850,35549,12854,18951,5895,13581,501115  
1860,813469,279122,23903,23933,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1385000,718712,68541,5653,51980,33091  
1900,1850993,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,44607,72021,11651,50148  
1930,26671123,2079128,44607,72021,11651,50148  
1940,1889924,2699285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1696010,1696010,1696010,1696010,1696010,781984  
1970,1539231,1465701,1471701,1471701,1471701,768460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,419782,8175133  
2015,1444018,2436733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

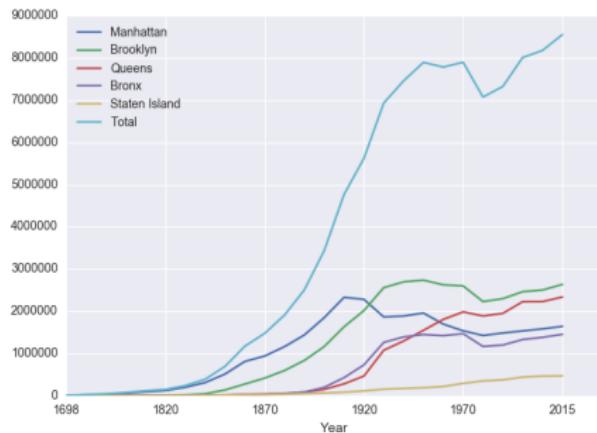
```
import matplotlib.pyplot as plt  
import pandas as pd
```

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

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Borough,Population  
1699,Manhattan, Brooklyn, Queens, Bronx, Staten Island, Total  
1771,21843,36231,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70531,6354,7041,1801,4937,93734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14041,5348,10965,391114  
1850,355441,21800,18500,5850,13000,45115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33051,1911801  
1890,1375000,718000,68000,63000,41000,215134  
1900,1850993,116582,152999,200567,67921,2437202  
1910,233142,1634351,2841,430980,8569,476683  
1920,2281103,2018354,44607,73201,11651,50048  
1930,2667103,2485254,579128,125245,15837,693446  
1940,1889924,2690285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7091957  
1960,1696000,27319,180900,1451277,191555,7091957  
1970,1539231,346570,2471473,1472701,135443,708460  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419728,8080879  
2010,1583873,2504705,2216722,1385108,451581,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6



Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 8: function parameters, github

- Functions can have **input parameters**.

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

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



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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

Python & Circuits Review: 10 Weeks in 10 Minutes



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

Lecture Quiz

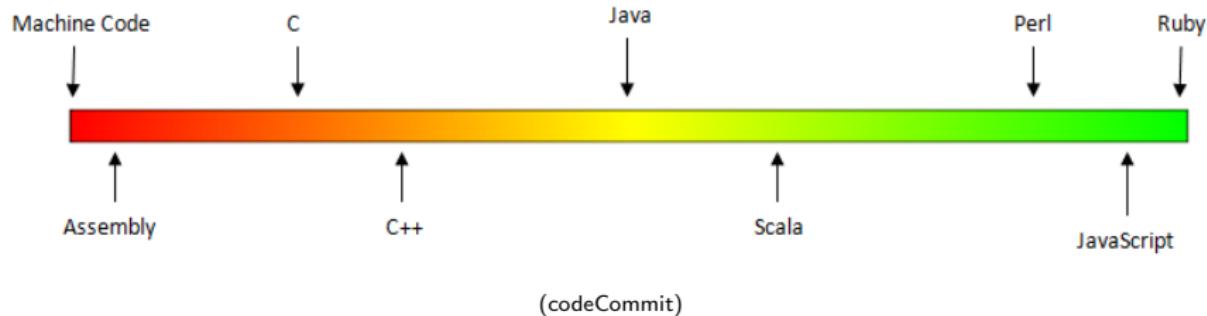
- Log-in to Gradescope
- Find LECTURE 11 Quiz
- Take the quiz
- **You have 3 minutes**

Today's Topics



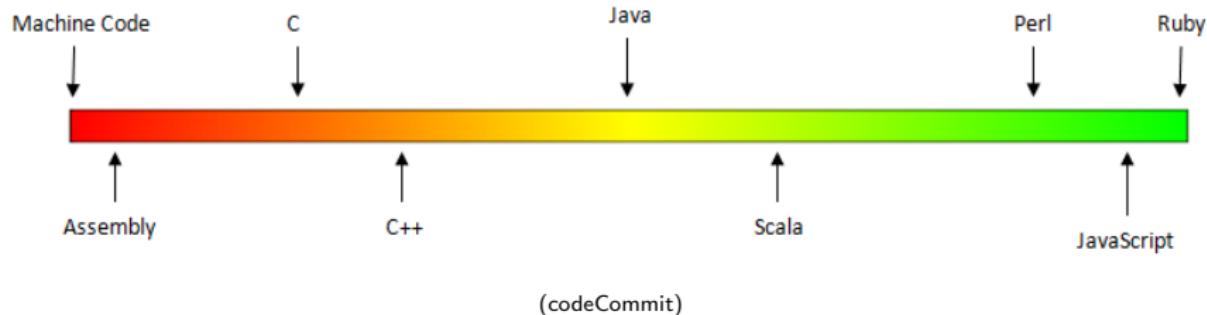
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

Low-Level vs. High-Level Languages



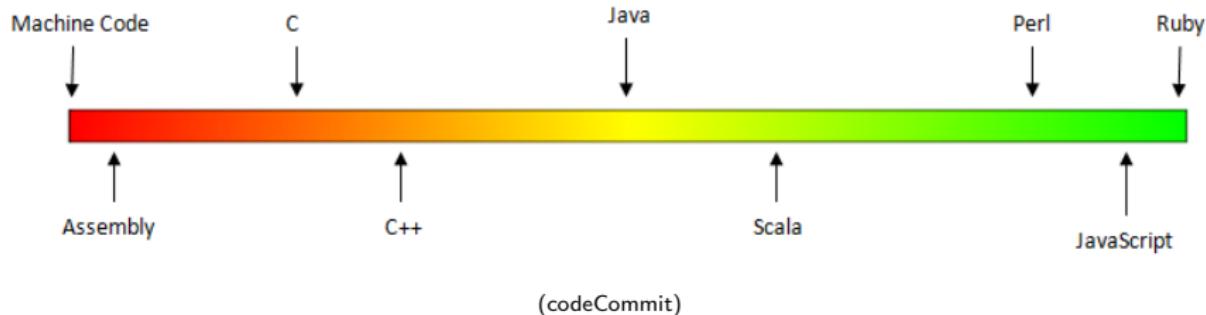
- Can view programming languages on a continuum.

Low-Level vs. High-Level Languages



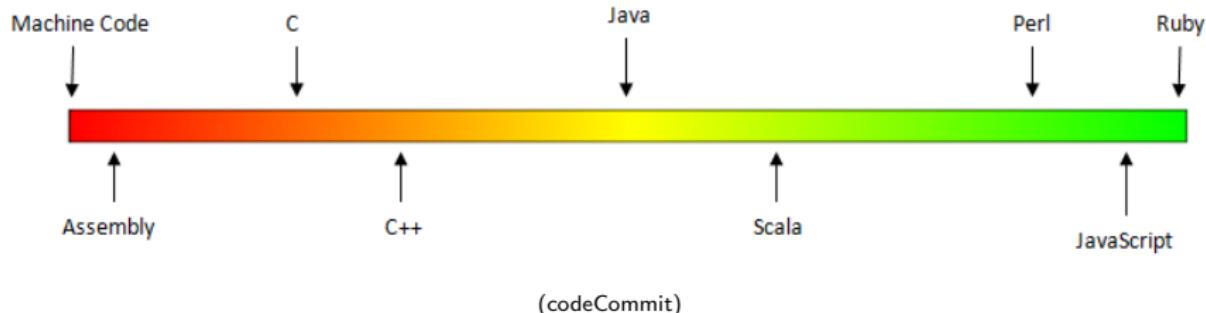
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

Low-Level vs. High-Level Languages



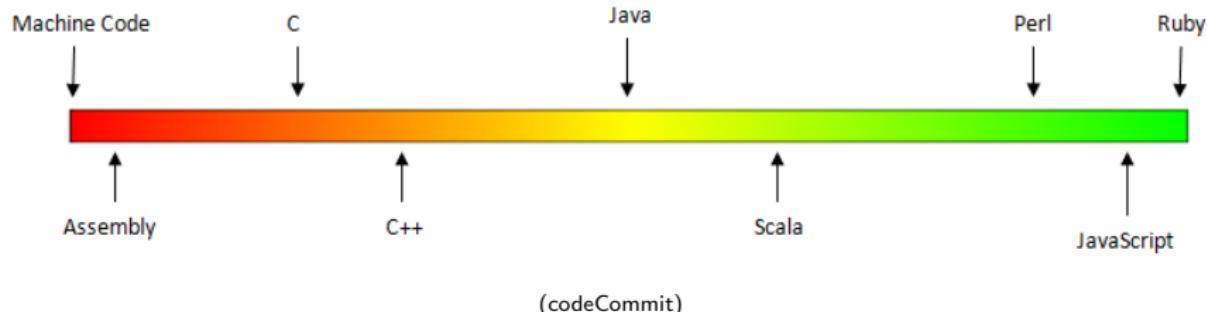
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

Low-Level vs. High-Level Languages



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

Low-Level vs. High-Level Languages



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

Processing

Bei [Lernzettel](#) für [Blindtext](#). "Blindtext" ist ein Dokument, das aussieht wie normaler Text, aber nur aus Ziffern besteht. Es kann nicht gelesen werden, ohne es zuvor ausdrucken oder es mit einem Blindenschriftleser zu öffnen.

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

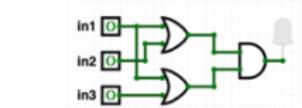


Data
&
Instructions

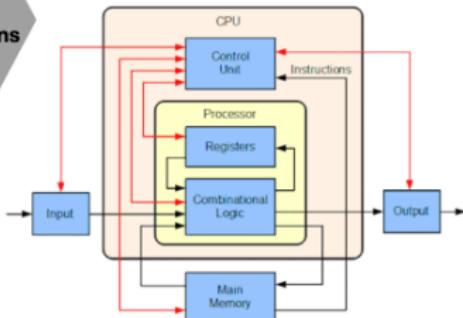
```
01110100011100100110000101  
1110010001101101110101011  
00111110001101000 101011  
00101110 00010000111111  
1110110111101000111011011  
001011010 101101 10011000  
01100101011010001000001  
11100100001101 00111101  
011000101101100011010101  
01000100000011000100 000  
0110010101 100110100101  
10010 100000110110011101  
01101100110010001101  
011000110 101110111001100  
01100110 10111011101101  
100 01 00 101100100100000  
0110011001000000110 1001  
1001011001000000110 1001  
011011001100100011010101  
011101100110011011101100  
11100110 0011011011101101  
01110010010000001101  
0010 01000 00100 100 101  
001000001110100011011100  
00010 110010011 100100100  
111010001011011011011 0100  
1 11011011100111011001001  
0111001001011100 10001101  
00000110110001 10110 0011
```

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

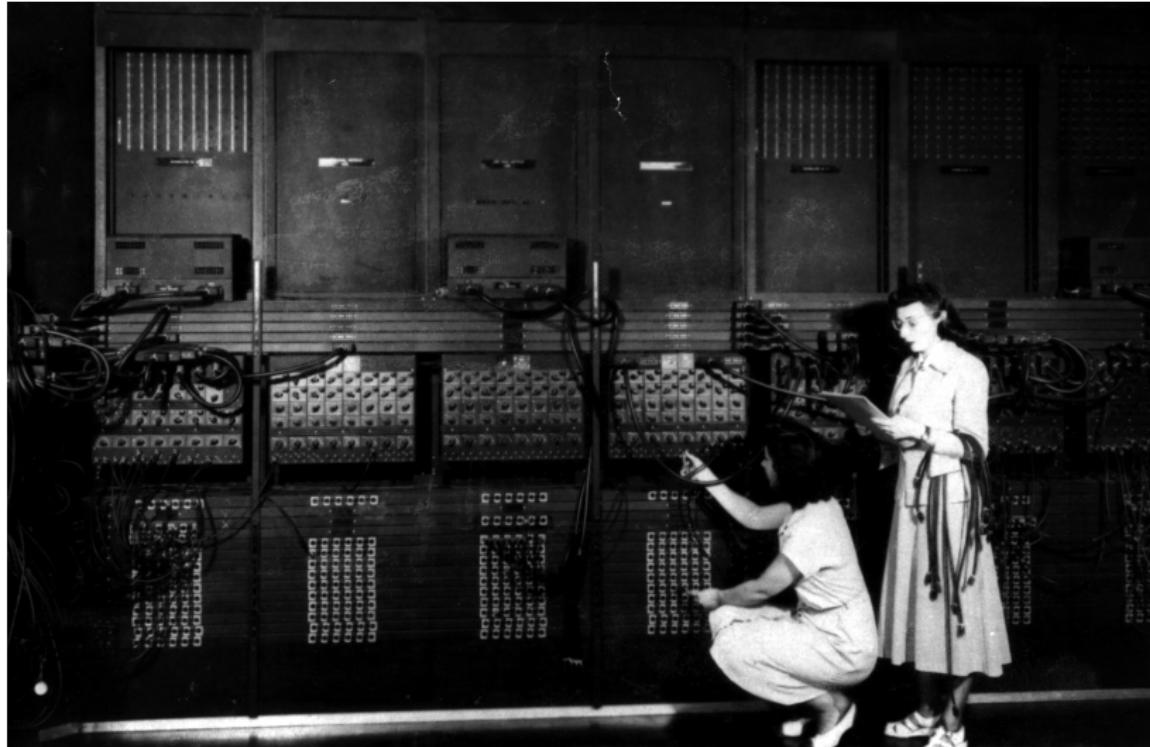
Data
&
Instructions



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



Machine Language



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

Machine Language

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

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

BREAK

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

(wiki)

Machine Language

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

(wiki)

Machine Language

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

(wiki)

Machine Language



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

Assembly Instruction	Binary Value
LDI R0, 1000	00 0000 C0 30
LDI R1, 1000	00 0000 C0 30
ADD R0, R1, R0	00 0002 1B
ADD R0, R1, R0	00 0002 1B
LDI R2, 1000	00 0003 FB
LDI R2, 1000	00 0003 FB
MUL R0, R1, R2	00 0004 34 12
MUL R0, R1, R2	00 0004 34 12
LDI R3, 1000	00 0005 69 21 43
LDI R3, 1000	00 0005 69 21 43
ADD R0, R1, R3	00 0006 40 C0 #4324
ADD R0, R1, R3	00 0006 40 C0 #4324
LDI R4, 1000	00 0007 69 F0 01
LDI R4, 1000	00 0007 69 F0 01
STA R4, R0	00 0008 8F 03 7F 01
STA R4, R0	00 0008 8F 03 7F 01
LDI R5, 1000	00 0009 E0 30
LDI R5, 1000	00 0009 E0 30
ADD R0, R1, R5	00 000A 60 3F 80
ADD R0, R1, R5	00 000A 60 3F 80
LDI R6, 1000	00 000B 60 3F 80
LDI R6, 1000	00 000B 60 3F 80
ADD R0, R1, R6	00 000C 60 3F 80
ADD R0, R1, R6	00 000C 60 3F 80
HALT	00 000D 00 00 00 00
HALT	00 000D 00 00 00 00
BREAK	00 000E 00 00 00 00
BREAK	00 000E 00 00 00 00
PC = 0000000000000000	PC = 0000000000000000
REGISTERS = 00000000 00000000 00000000 00000000	REGISTERS = 00000000 00000000 00000000 00000000
DATA = 00000000 00000000 00000000 00000000	DATA = 00000000 00000000 00000000 00000000
Stack = 00000000 00000000 00000000 00000000	Stack = 00000000 00000000 00000000 00000000

(wiki)

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

Machine Language



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

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

(wiki)

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

"Hello World!" in Simplified Machine Language

Line: 3 Go!

Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

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

(WeMIPS)

WeMIPS

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

ADDI $t0, $zero, 4 # 4 is for print string
LD $t0, $t0, $t0, 0
JAL $t0, _print

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

ADDI $t0, $zero, 4 # 4 is for print string
LD $t0, $t0, $t0, 0
JAL $t0, _print
```

(Demo with WeMIPS)

MIPS Commands

The screenshot shows a MIPS assembly debugger interface. At the top, there's a menu bar with 'File', 'Edit', 'Run', 'Show/Hide Demo', 'Addition', 'Subtraction', 'Multiplication', 'Division', 'Hello World', 'Code Gen Base String', 'Interactive', 'Binary Decimal', 'Decimal Binary', and 'Debug'. Below the menu is a toolbar with 'Step', 'Run', 'Break', 'Create auto watching', and 'Stop' buttons. The main area contains assembly code and a register table.

Assembly Code:

```
1 # Ensure 'Hello, world!' is at the top of the stack
2 .text
3 .globl _start
4 .type _start, @function
5 _start:
6    li $t0, 11110000
7    li $t1, 11110001
8    sb $t0, $t1($sp)
9    addi $sp, $sp, -4
10   li $t0, 11110000
11   li $t1, 11110001
12   sb $t0, $t1($sp)
13   addi $sp, $sp, -4
14   addi $t0, $t0, 1111 # r
15   addi $t1, $t1, 1111 # r
16   add $t2, $t0, $t1
17   add $t3, $t2, $t0
18   add $t4, $t3, $t1
19   add $t5, $t4, $t0
20   add $t6, $t5, $t1
21   add $t7, $t6, $t0
22   add $t8, $t7, $t1
23   add $t9, $t8, $t0
24   add $t10, $t9, $t1
25   add $t11, $t10, $t0
26   add $t12, $t11, $t1
27   add $t13, $t12, $t0
28   add $t14, $t13, $t1
29   add $t15, $t14, $t0
30   add $t16, $t15, $t1
31   add $t17, $t16, $t0
32   add $t18, $t17, $t1
33   add $t19, $t18, $t0
34   add $t20, $t19, $t1
35   add $t21, $t20, $t0
36   add $t22, $t21, $t1
37   add $t23, $t22, $t0
38   add $t24, $t23, $t1
39   add $t25, $t24, $t0
40   add $t26, $t25, $t1
41   add $t27, $t26, $t0
42   add $t28, $t27, $t1
43   add $t29, $t28, $t0
44   add $t30, $t29, $t1
45   add $t31, $t30, $t0
46   add $t32, $t31, $t1
47   add $t33, $t32, $t0
48   add $t34, $t33, $t1
49   add $t35, $t34, $t0
50   add $t36, $t35, $t1
51   add $t37, $t36, $t0
52   add $t38, $t37, $t1
53   add $t39, $t38, $t0
54   add $t40, $t39, $t1
55   add $t41, $t40, $t0
56   add $t42, $t41, $t1
57   add $t43, $t42, $t0
58   add $t44, $t43, $t1
59   add $t45, $t44, $t0
60   add $t46, $t45, $t1
61   add $t47, $t46, $t0
62   add $t48, $t47, $t1
63   add $t49, $t48, $t0
64   add $t50, $t49, $t1
65   add $t51, $t50, $t0
66   add $t52, $t51, $t1
67   add $t53, $t52, $t0
68   add $t54, $t53, $t1
69   add $t55, $t54, $t0
70   add $t56, $t55, $t1
71   add $t57, $t56, $t0
72   add $t58, $t57, $t1
73   add $t59, $t58, $t0
74   add $t60, $t59, $t1
75   add $t61, $t60, $t0
76   add $t62, $t61, $t1
77   add $t63, $t62, $t0
78   add $t64, $t63, $t1
79   add $t65, $t64, $t0
80   add $t66, $t65, $t1
81   add $t67, $t66, $t0
82   add $t68, $t67, $t1
83   add $t69, $t68, $t0
84   add $t70, $t69, $t1
85   add $t71, $t70, $t0
86   add $t72, $t71, $t1
87   add $t73, $t72, $t0
88   add $t74, $t73, $t1
89   add $t75, $t74, $t0
90   add $t76, $t75, $t1
91   add $t77, $t76, $t0
92   add $t78, $t77, $t1
93   add $t79, $t78, $t0
94   add $t80, $t79, $t1
95   add $t81, $t80, $t0
96   add $t82, $t81, $t1
97   add $t83, $t82, $t0
98   add $t84, $t83, $t1
99   add $t85, $t84, $t0
100  add $t86, $t85, $t1
101  add $t87, $t86, $t0
102  add $t88, $t87, $t1
103  add $t89, $t88, $t0
104  add $t90, $t89, $t1
105  add $t91, $t90, $t0
106  add $t92, $t91, $t1
107  add $t93, $t92, $t0
108  add $t94, $t93, $t1
109  add $t95, $t94, $t0
110  add $t96, $t95, $t1
111  add $t97, $t96, $t0
112  add $t98, $t97, $t1
113  add $t99, $t98, $t0
114  add $t100, $t99, $t1
115  add $t101, $t100, $t0
116  add $t102, $t101, $t1
117  add $t103, $t102, $t0
118  add $t104, $t103, $t1
119  add $t105, $t104, $t0
120  add $t106, $t105, $t1
121  add $t107, $t106, $t0
122  add $t108, $t107, $t1
123  add $t109, $t108, $t0
124  add $t110, $t109, $t1
125  add $t111, $t110, $t0
126  add $t112, $t111, $t1
127  add $t113, $t112, $t0
128  add $t114, $t113, $t1
129  add $t115, $t114, $t0
130  add $t116, $t115, $t1
131  add $t117, $t116, $t0
132  add $t118, $t117, $t1
133  add $t119, $t118, $t0
134  add $t120, $t119, $t1
135  add $t121, $t120, $t0
136  add $t122, $t121, $t1
137  add $t123, $t122, $t0
138  add $t124, $t123, $t1
139  add $t125, $t124, $t0
140  add $t126, $t125, $t1
141  add $t127, $t126, $t0
142  add $t128, $t127, $t1
143  add $t129, $t128, $t0
144  add $t130, $t129, $t1
145  add $t131, $t130, $t0
146  add $t132, $t131, $t1
147  add $t133, $t132, $t0
148  add $t134, $t133, $t1
149  add $t135, $t134, $t0
150  add $t136, $t135, $t1
151  add $t137, $t136, $t0
152  add $t138, $t137, $t1
153  add $t139, $t138, $t0
154  add $t140, $t139, $t1
155  add $t141, $t140, $t0
156  add $t142, $t141, $t1
157  add $t143, $t142, $t0
158  add $t144, $t143, $t1
159  add $t145, $t144, $t0
160  add $t146, $t145, $t1
161  add $t147, $t146, $t0
162  add $t148, $t147, $t1
163  add $t149, $t148, $t0
164  add $t150, $t149, $t1
165  add $t151, $t150, $t0
166  add $t152, $t151, $t1
167  add $t153, $t152, $t0
168  add $t154, $t153, $t1
169  add $t155, $t154, $t0
170  add $t156, $t155, $t1
171  add $t157, $t156, $t0
172  add $t158, $t157, $t1
173  add $t159, $t158, $t0
174  add $t160, $t159, $t1
175  add $t161, $t160, $t0
176  add $t162, $t161, $t1
177  add $t163, $t162, $t0
178  add $t164, $t163, $t1
179  add $t165, $t164, $t0
180  add $t166, $t165, $t1
181  add $t167, $t166, $t0
182  add $t168, $t167, $t1
183  add $t169, $t168, $t0
184  add $t170, $t169, $t1
185  add $t171, $t170, $t0
186  add $t172, $t171, $t1
187  add $t173, $t172, $t0
188  add $t174, $t173, $t1
189  add $t175, $t174, $t0
190  add $t176, $t175, $t1
191  add $t177, $t176, $t0
192  add $t178, $t177, $t1
193  add $t179, $t178, $t0
194  add $t180, $t179, $t1
195  add $t181, $t180, $t0
196  add $t182, $t181, $t1
197  add $t183, $t182, $t0
198  add $t184, $t183, $t1
199  add $t185, $t184, $t0
200  add $t186, $t185, $t1
201  add $t187, $t186, $t0
202  add $t188, $t187, $t1
203  add $t189, $t188, $t0
204  add $t190, $t189, $t1
205  add $t191, $t190, $t0
206  add $t192, $t191, $t1
207  add $t193, $t192, $t0
208  add $t194, $t193, $t1
209  add $t195, $t194, $t0
210  add $t196, $t195, $t1
211  add $t197, $t196, $t0
212  add $t198, $t197, $t1
213  add $t199, $t198, $t0
214  add $t200, $t199, $t1
215  add $t201, $t200, $t0
216  add $t202, $t201, $t1
217  add $t203, $t202, $t0
218  add $t204, $t203, $t1
219  add $t205, $t204, $t0
220  add $t206, $t205, $t1
221  add $t207, $t206, $t0
222  add $t208, $t207, $t1
223  add $t209, $t208, $t0
224  add $t210, $t209, $t1
225  add $t211, $t210, $t0
226  add $t212, $t211, $t1
227  add $t213, $t212, $t0
228  add $t214, $t213, $t1
229  add $t215, $t214, $t0
230  add $t216, $t215, $t1
231  add $t217, $t216, $t0
232  add $t218, $t217, $t1
233  add $t219, $t218, $t0
234  add $t220, $t219, $t1
235  add $t221, $t220, $t0
236  add $t222, $t221, $t1
237  add $t223, $t222, $t0
238  add $t224, $t223, $t1
239  add $t225, $t224, $t0
240  add $t226, $t225, $t1
241  add $t227, $t226, $t0
242  add $t228, $t227, $t1
243  add $t229, $t228, $t0
244  add $t230, $t229, $t1
245  add $t231, $t230, $t0
246  add $t232, $t231, $t1
247  add $t233, $t232, $t0
248  add $t234, $t233, $t1
249  add $t235, $t234, $t0
250  add $t236, $t235, $t1
251  add $t237, $t236, $t0
252  add $t238, $t237, $t1
253  add $t239, $t238, $t0
254  add $t240, $t239, $t1
255  add $t241, $t240, $t0
256  add $t242, $t241, $t1
257  add $t243, $t242, $t0
258  add $t244, $t243, $t1
259  add $t245, $t244, $t0
260  add $t246, $t245, $t1
261  add $t247, $t246, $t0
262  add $t248, $t247, $t1
263  add $t249, $t248, $t0
264  add $t250, $t249, $t1
265  add $t251, $t250, $t0
266  add $t252, $t251, $t1
267  add $t253, $t252, $t0
268  add $t254, $t253, $t1
269  add $t255, $t254, $t0
270  add $t256, $t255, $t1
271  add $t257, $t256, $t0
272  add $t258, $t257, $t1
273  add $t259, $t258, $t0
274  add $t260, $t259, $t1
275  add $t261, $t260, $t0
276  add $t262, $t261, $t1
277  add $t263, $t262, $t0
278  add $t264, $t263, $t1
279  add $t265, $t264, $t0
280  add $t266, $t265, $t1
281  add $t267, $t266, $t0
282  add $t268, $t267, $t1
283  add $t269, $t268, $t0
284  add $t270, $t269, $t1
285  add $t271, $t270, $t0
286  add $t272, $t271, $t1
287  add $t273, $t272, $t0
288  add $t274, $t273, $t1
289  add $t275, $t274, $t0
290  add $t276, $t275, $t1
291  add $t277, $t276, $t0
292  add $t278, $t277, $t1
293  add $t279, $t278, $t0
294  add $t280, $t279, $t1
295  add $t281, $t280, $t0
296  add $t282, $t281, $t1
297  add $t283, $t282, $t0
298  add $t284, $t283, $t1
299  add $t285, $t284, $t0
300  add $t286, $t285, $t1
301  add $t287, $t286, $t0
302  add $t288, $t287, $t1
303  add $t289, $t288, $t0
304  add $t290, $t289, $t1
305  add $t291, $t290, $t0
306  add $t292, $t291, $t1
307  add $t293, $t292, $t0
308  add $t294, $t293, $t1
309  add $t295, $t294, $t0
310  add $t296, $t295, $t1
311  add $t297, $t296, $t0
312  add $t298, $t297, $t1
313  add $t299, $t298, $t0
314  add $t300, $t299, $t1
315  add $t301, $t300, $t0
316  add $t302, $t301, $t1
317  add $t303, $t302, $t0
318  add $t304, $t303, $t1
319  add $t305, $t304, $t0
320  add $t306, $t305, $t1
321  add $t307, $t306, $t0
322  add $t308, $t307, $t1
323  add $t309, $t308, $t0
324  add $t310, $t309, $t1
325  add $t311, $t310, $t0
326  add $t312, $t311, $t1
327  add $t313, $t312, $t0
328  add $t314, $t313, $t1
329  add $t315, $t314, $t0
330  add $t316, $t315, $t1
331  add $t317, $t316, $t0
332  add $t318, $t317, $t1
333  add $t319, $t318, $t0
334  add $t320, $t319, $t1
335  add $t321, $t320, $t0
336  add $t322, $t321, $t1
337  add $t323, $t322, $t0
338  add $t324, $t323, $t1
339  add $t325, $t324, $t0
340  add $t326, $t325, $t1
341  add $t327, $t326, $t0
342  add $t328, $t327, $t1
343  add $t329, $t328, $t0
344  add $t330, $t329, $t1
345  add $t331, $t330, $t0
346  add $t332, $t331, $t1
347  add $t333, $t332, $t0
348  add $t334, $t333, $t1
349  add $t335, $t334, $t0
350  add $t336, $t335, $t1
351  add $t337, $t336, $t0
352  add $t338, $t337, $t1
353  add $t339, $t338, $t0
354  add $t340, $t339, $t1
355  add $t341, $t340, $t0
356  add $t342, $t341, $t1
357  add $t343, $t342, $t0
358  add $t344, $t343, $t1
359  add $t345, $t344, $t0
360  add $t346, $t345, $t1
361  add $t347, $t346, $t0
362  add $t348, $t347, $t1
363  add $t349, $t348, $t0
364  add $t350, $t349, $t1
365  add $t351, $t350, $t0
366  add $t352, $t351, $t1
367  add $t353, $t352, $t0
368  add $t354, $t353, $t1
369  add $t355, $t354, $t0
370  add $t356, $t355, $t1
371  add $t357, $t356, $t0
372  add $t358, $t357, $t1
373  add $t359, $t358, $t0
374  add $t360, $t359, $t1
375  add $t361, $t360, $t0
376  add $t362, $t361, $t1
377  add $t363, $t362, $t0
378  add $t364, $t363, $t1
379  add $t365, $t364, $t0
380  add $t366, $t365, $t1
381  add $t367, $t366, $t0
382  add $t368, $t367, $t1
383  add $t369, $t368, $t0
384  add $t370, $t369, $t1
385  add $t371, $t370, $t0
386  add $t372, $t371, $t1
387  add $t373, $t372, $t0
388  add $t374, $t373, $t1
389  add $t375, $t374, $t0
390  add $t376, $t375, $t1
391  add $t377, $t376, $t0
392  add $t378, $t377, $t1
393  add $t379, $t378, $t0
394  add $t380, $t379, $t1
395  add $t381, $t380, $t0
396  add $t382, $t381, $t1
397  add $t383, $t382, $t0
398  add $t384, $t383, $t1
399  add $t385, $t384, $t0
400  add $t386, $t385, $t1
401  add $t387, $t386, $t0
402  add $t388, $t387, $t1
403  add $t389, $t388, $t0
404  add $t390, $t389, $t1
405  add $t391, $t390, $t0
406  add $t392, $t391, $t1
407  add $t393, $t392, $t0
408  add $t394, $t393, $t1
409  add $t395, $t394, $t0
410  add $t396, $t395, $t1
411  add $t397, $t396, $t0
412  add $t398, $t397, $t1
413  add $t399, $t398, $t0
414  add $t400, $t399, $t1
415  add $t401, $t400, $t0
416  add $t402, $t401, $t1
417  add $t403, $t402, $t0
418  add $t404, $t403, $t1
419  add $t405, $t404, $t0
420  add $t406, $t405, $t1
421  add $t407, $t406, $t0
422  add $t408, $t407, $t1
423  add $t409, $t408, $t0
424  add $t410, $t409, $t1
425  add $t411, $t410, $t0
426  add $t412, $t411, $t1
427  add $t413, $t412, $t0
428  add $t414, $t413, $t1
429  add $t415, $t414, $t0
430  add $t416, $t415, $t1
431  add $t417, $t416, $t0
432  add $t418, $t417, $t1
433  add $t419, $t418, $t0
434  add $t420, $t419, $t1
435  add $t421, $t420, $t0
436  add $t422, $t421, $t1
437  add $t423, $t422, $t0
438  add $t424, $t423, $t1
439  add $t425, $t424, $t0
440  add $t426, $t425, $t1
441  add $t427, $t426, $t0
442  add $t428, $t427, $t1
443  add $t429, $t428, $t0
444  add $t430, $t429, $t1
445  add $t431, $t430, $t0
446  add $t432, $t431, $t1
447  add $t433, $t432, $t0
448  add $t434, $t433, $t1
449  add $t435, $t434, $t0
450  add $t436, $t435, $t1
451  add $t437, $t436, $t0
452  add $t438, $t437, $t1
453  add $t439, $t438, $t0
454  add $t440, $t439, $t1
455  add $t441, $t440, $t0
456  add $t442, $t441, $t1
457  add $t443, $t442, $t0
458  add $t444, $t443, $t1
459  add $t445, $t444, $t0
460  add $t446, $t445, $t1
461  add $t447, $t446, $t0
462  add $t448, $t447, $t1
463  add $t449, $t448, $t0
464  add $t450, $t449, $t1
465  add $t451, $t450, $t0
466  add $t452, $t451, $t1
467  add $t453, $t452, $t0
468  add $t454, $t453, $t1
469  add $t455, $t454, $t0
470  add $t456, $t455, $t1
471  add $t457, $t456, $t0
472  add $t458, $t457, $t1
473  add $t459, $t458, $t0
474  add $t460, $t459, $t1
475  add $t461, $t460, $t0
476  add $t462, $t461, $t1
477  add $t463, $t462, $t0
478  add $t464, $t463, $t1
479  add $t465, $t464, $t0
480  add $t466, $t465, $t1
481  add $t467, $t466, $t0
482  add $t468, $t467, $t1
483  add $t469, $t468, $t0
484  add $t470, $t469, $t1
485  add $t471, $t470, $t0
486  add $t472, $t471, $t1
487  add $t473, $t472, $t0
488  add $t474, $t473, $t1
489  add $t475, $t474, $t0
490  add $t476, $t475, $t1
491  add $t477, $t476, $t0
492  add $t478, $t477, $t1
493  add $t479, $t478, $t0
494  add $t480, $t479, $t1
495  add $t481, $t480, $t0
496  add $t482, $t481, $t1
497  add $t483, $t482, $t0
498  add $t484, $t483, $t1
499  add $t485, $t484, $t0
500  add $t486, $t485, $t1
501  add $t487, $t486, $t0
502  add $t488, $t487, $t1
503  add $t489, $t488, $t0
504  add $t490, $t489, $t1
505  add $t491, $t490, $t0
506  add $t492, $t491, $t1
507  add $t493, $t492, $t0
508  add $t494, $t493, $t1
509  add $t495, $t494, $t0
510  add $t496, $t495, $t1
511  add $t497, $t496, $t0
512  add $t498, $t497, $t1
513  add $t499, $t498, $t0
514  add $t500, $t499, $t1
515  add $t501, $t500, $t0
516  add $t502, $t501, $t1
517  add $t503, $t502, $t0
518  add $t504, $t503, $t1
519  add $t505, $t504, $t0
520  add $t506, $t505, $t1
521  add $t507, $t506, $t0
522  add $t508, $t507, $t1
523  add $t509, $t508, $t0
524  add $t510, $t509, $t1
525  add $t511, $t510, $t0
526  add $t512, $t511, $t1
527  add $t513, $t512, $t0
528  add $t514, $t513, $t1
529  add $t515, $t514, $t0
530  add $t516, $t515, $t1
531  add $t517, $t516, $t0
532  add $t518, $t517, $t1
533  add $t519, $t518, $t0
534  add $t520, $t519, $t1
535  add $t521, $t520, $t0
536  add $t522, $t521, $t1
537  add $t523, $t522, $t0
538  add $t524, $t523, $t1
539  add $t525, $t524, $t0
540  add $t526, $t525, $t1
541  add $t527, $t526, $t0
542  add $t528, $t527, $t1
543  add $t529, $t528, $t0
544  add $t530, $t529, $t1
545  add $t531, $t530, $t0
546  add $t532, $t531, $t1
547  add $t533, $t532, $t0
548  add $t534, $t533, $t1
549  add $t535, $t534, $t0
550  add $t536, $t535, $t1
551  add $t537, $t536, $t0
552  add $t538, $t537, $t1
553  add $t539, $t538, $t0
554  add $t540, $t539, $t1
555  add $t541, $t540, $t0
556  add $t542, $t541, $t1
557  add $t543, $t542, $t0
558  add $t544, $t543, $t1
559  add $t545, $t544, $t0
560  add $t546, $t545, $t1
561  add $t547, $t546, $t0
562  add $t548, $t547, $t1
563  add $t549, $t548, $t0
564  add $t550, $t549, $t1
565  add $t551, $t550, $t0
566  add $t552, $t551, $t1
567  add $t553, $t552, $t0
568  add $t554, $t553, $t1
569  add $t555, $t554, $t0
570  add $t556, $t555, $t1
571  add $t557, $t556, $t0
572  add $t558, $t557, $t1
573  add $t559, $t558, $t0
574  add $t560, $t559, $t1
575  add $t561, $t560, $t0
576  add $t562, $t561, $t1
577  add $t563, $t562, $t0
578  add $t564, $t563, $t1
579  add $t565, $t564, $t0
580  add $t566, $t565, $t1
581  add $t567, $t566, $t0
582  add $t568, $t567, $t1
583  add $t569, $t568, $t0
584  add $t570, $t569, $t1
585  add $t571, $t570, $t0
586  add $t572, $t571, $t1
587  add $t573, $t572, $t0
588  add $t574, $t573, $t1
589  add $t575, $t574, $t0
590  add $t576, $t575, $t1
591  add $t577, $t576, $t0
592  add $t578, $t577, $t1
593  add $t579, $t578, $t0
594  add $t580, $t579, $t1
595  add $t581, $t580, $t0
596  add $t582, $t581, $t1
597  add $t583, $t582, $t0
598  add $t584, $t583, $t1
599  add $t585, $t584, $t0
600
```

MIPS Commands

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

MIPS Commands

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

MIPS Commands

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

MIPS Commands

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

MIPS Commands

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

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

MIPS Commands

```
Line 3 Set Showchee Demos User Guide | List Tasks | Close
Additional Decoder Hex Loader Stack Test Help Window
Code Gen Save String Interactive Binary Decimal Decimal/Hexadecimal
Debug

d Shows the local code at the top of the stack
d0 00401000 00401000 $0000 0000000000000000
d1 00401000 00401000 $0000 0000000000000000
d2 00401000 00401000 $0000 0000000000000000
d3 00401000 00401000 $0000 0000000000000000
d4 00401000 00401000 $0000 0000000000000000
d5 00401000 00401000 $0000 0000000000000000
d6 00401000 00401000 $0000 0000000000000000
d7 00401000 00401000 $0000 0000000000000000
d8 00401000 00401000 $0000 0000000000000000
d9 00401000 00401000 $0000 0000000000000000
dA 00401000 00401000 $0000 0000000000000000
dB 00401000 00401000 $0000 0000000000000000
dC 00401000 00401000 $0000 0000000000000000
dD 00401000 00401000 $0000 0000000000000000
dE 00401000 00401000 $0000 0000000000000000
dF 00401000 00401000 $0000 0000000000000000
Step Run ✓ Enable auto switching
S T A V Stack Log
$0 10
$1 9
$0 8
$0 20
$0 656
$0 676
$0 807
$0 418

d ADDC $0, $0, $0, # $0 is for print string
d ADDC $0, $0, $0, # print to the log
$0x00401000 # points to the log
```

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

MIPS Commands

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

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

MIPS Commands

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

Challenge:

Line: 3 Go! Show/Hide Demos

User Guide | Unit Tests | Docs

Addition Doubler Stav Looper Stack Test Hello World

Code Gen Save String Interactive Binary2 Decimal Decimal2 Binary

Debug

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

Step Run Enable auto switching

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

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

WeMIPS

(Demo with WeMIPS)

Today's Topics



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

Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Loops & Jumps in Machine Language

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



Jump Demo

Line: 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

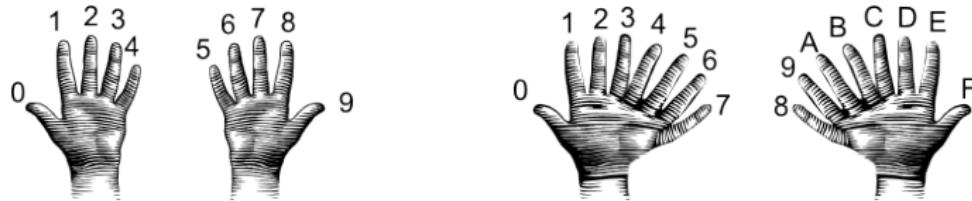


Today's Topics



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

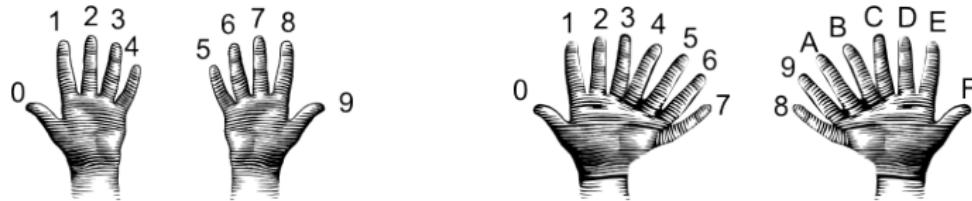
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

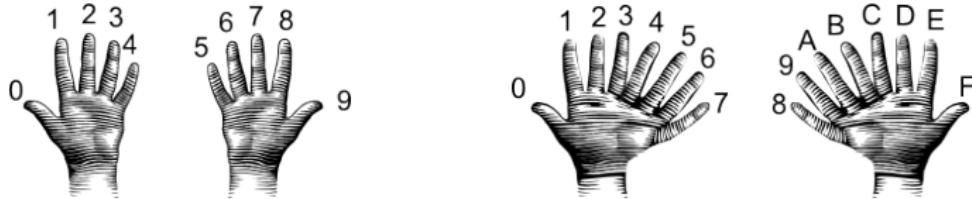
Hexadecimal to Decimal: Converting Between Bases



(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.

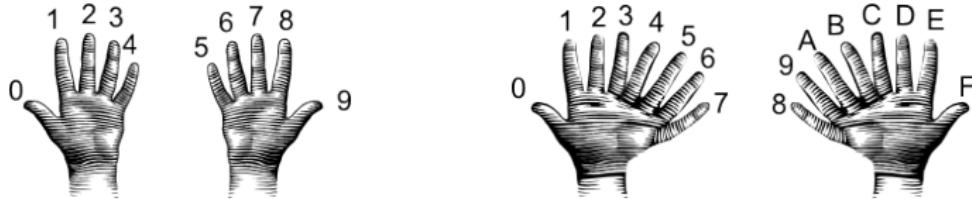
Hexadecimal to Decimal: Converting Between Bases



(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?

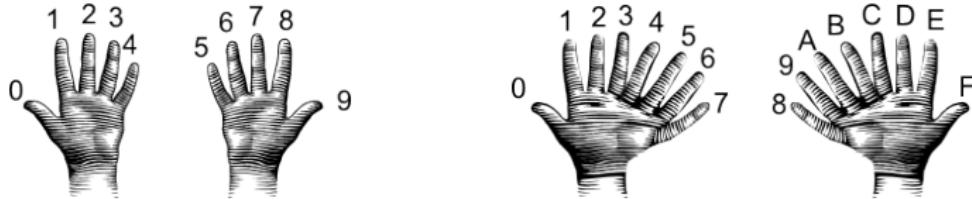
Hexadecimal to Decimal: Converting Between Bases



(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.

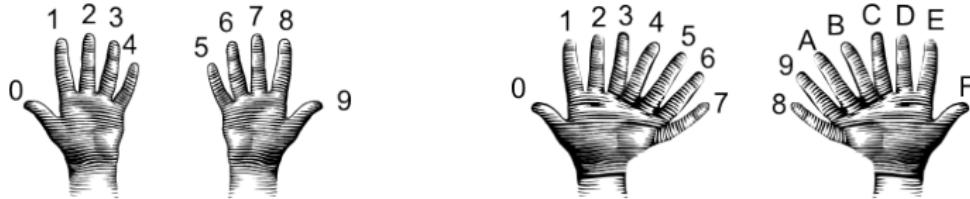
Hexadecimal to Decimal: Converting Between Bases



(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.

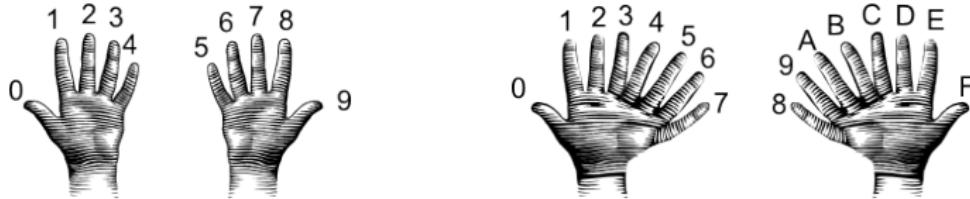
Hexadecimal to Decimal: Converting Between Bases



(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):

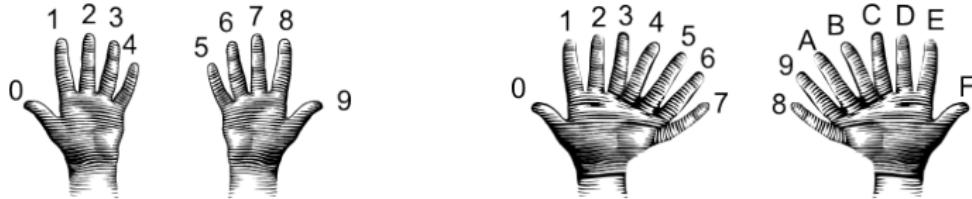
- Convert first digit to decimal and multiple by 16.
 - Convert second digit to decimal and add to total.
 - Example: what is 2A as a decimal number?

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

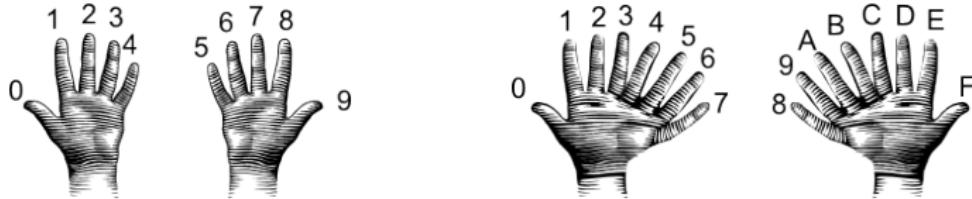
Hexadecimal to Decimal: Converting Between Bases



(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?

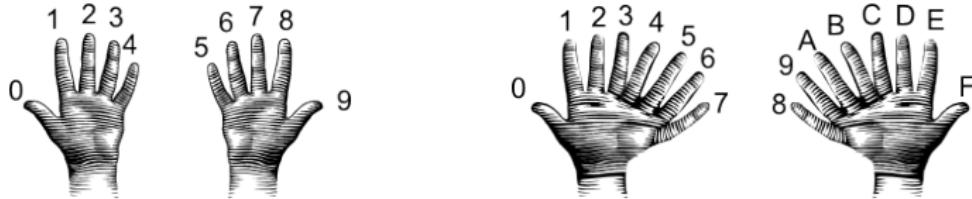
Hexadecimal to Decimal: Converting Between Bases



(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.

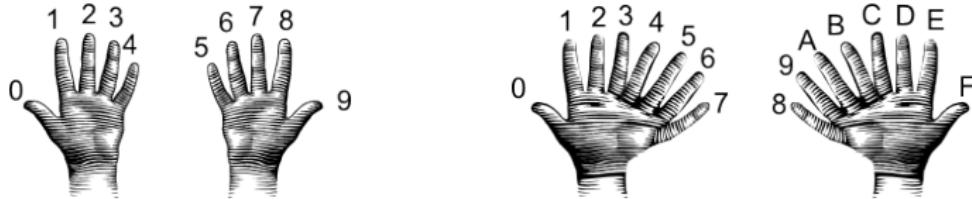
Hexadecimal to Decimal: Converting Between Bases



(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.

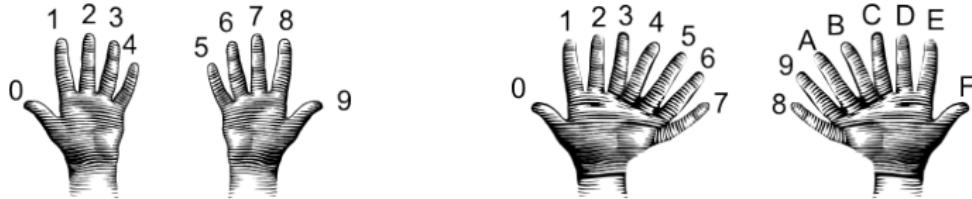
Hexadecimal to Decimal: Converting Between Bases



(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.

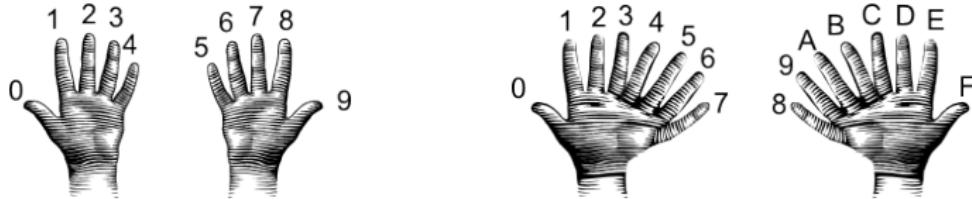
- Example: what is 99 as a decimal number?

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?

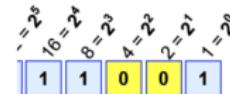
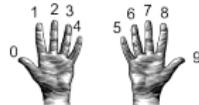
9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Answer is 153.

Decimal to Binary: Converting Between Bases

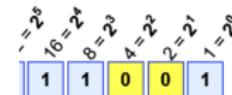
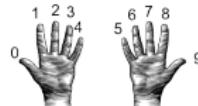


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

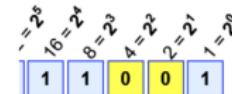
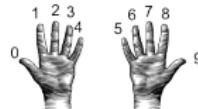


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

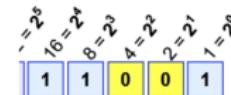
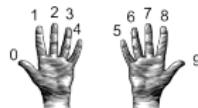


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

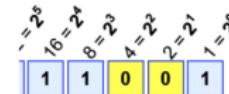
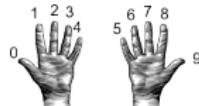


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

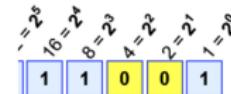


$$\text{Example: } 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

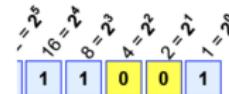
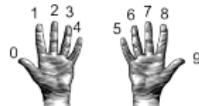


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

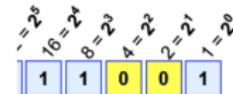
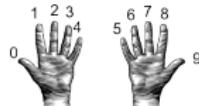


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

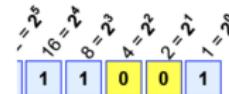
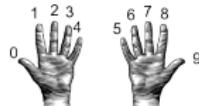


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

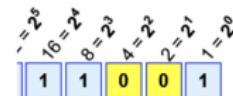
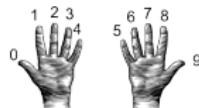


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases



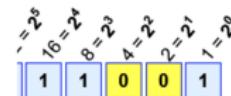
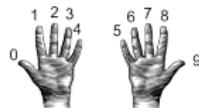
- From decimal to binary:

- 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



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

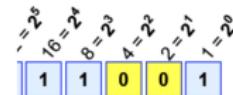
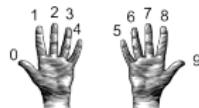
- From decimal to binary:

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

► Example: what is 130 in binary notation?

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

Decimal to Binary: Converting Between Bases



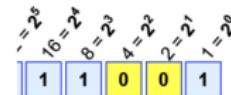
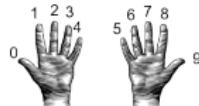
- From decimal to binary:

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

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

2/64 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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

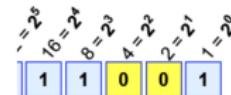
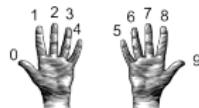
- From decimal to binary:

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

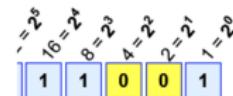
- From decimal to binary:

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

- From decimal to binary:

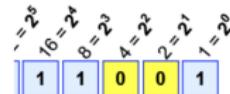
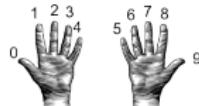
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

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

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

2/32 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

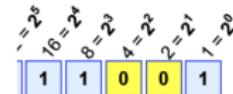
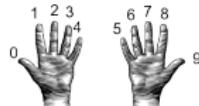
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

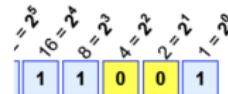
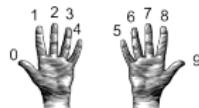
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

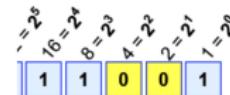
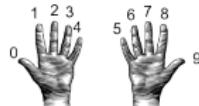
130/128 is 1 rem 2. First digit is 1: 1...

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

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

2/16 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

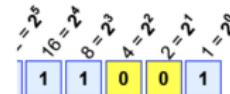
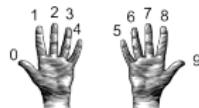
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

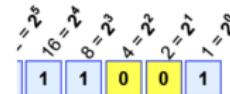
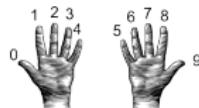
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

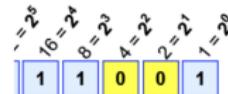
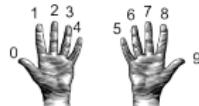
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

2/8 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

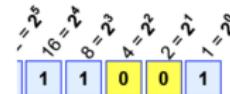
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

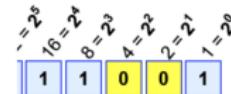
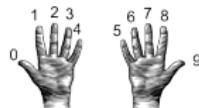
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

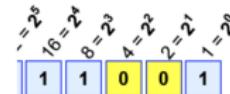
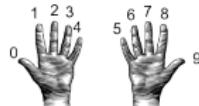
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

2/4 is 0 remainder 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

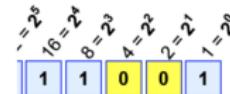
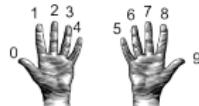
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

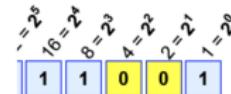
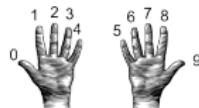
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

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

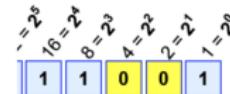
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

2/2 is 1 rem 0.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

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

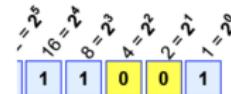
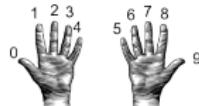
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

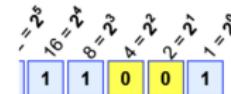
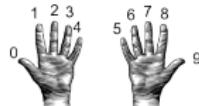
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

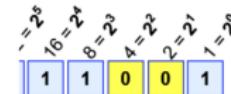
2/8 is 0 rem 2. Next digit is 0: 10000...

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



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

- From decimal to binary:

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

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

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

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

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

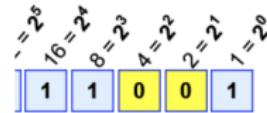
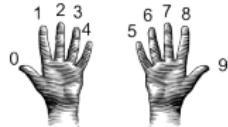
2/8 is 0 rem 2. Next digit is 0: 10000...

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

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

Adding the last remainder: 10000010

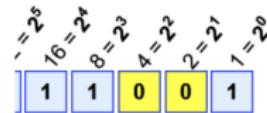
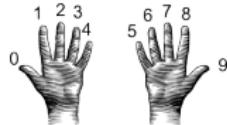
Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

Decimal to Binary: Converting Between Bases

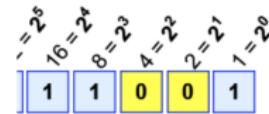


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99 / 128$ is 0 rem 99.

Decimal to Binary: Converting Between Bases

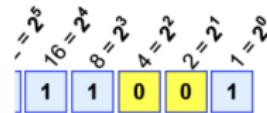


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases



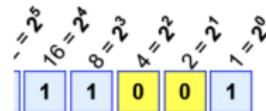
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



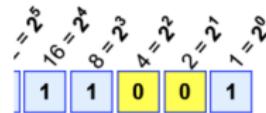
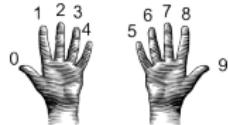
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



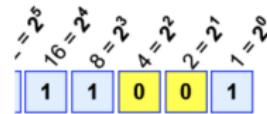
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

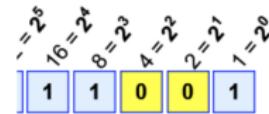
- Example: what is 99 in binary notation?

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

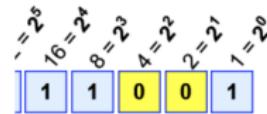
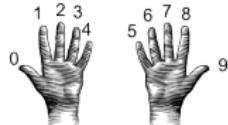
- Example: what is 99 in binary notation?

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

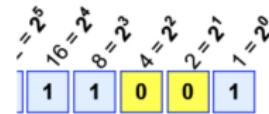
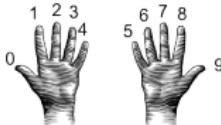
- Example: what is 99 in binary notation?

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

- Example: what is 99 in binary notation?

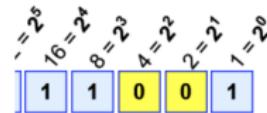
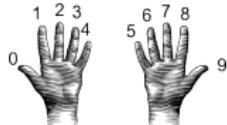
99/128 is 0 rem 99. First digit is 0: 0...

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

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

3/16 is 0 rem 3.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

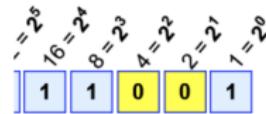
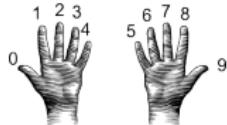
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

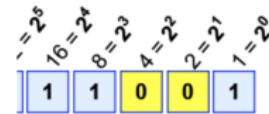
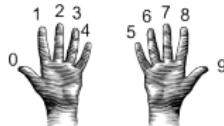
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

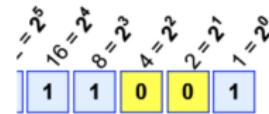
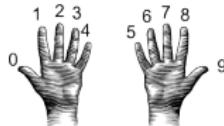
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

3/8 is 0 rem 3.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

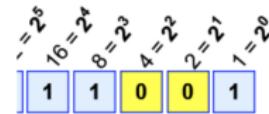
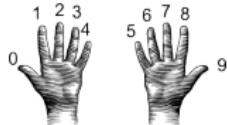
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

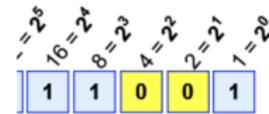
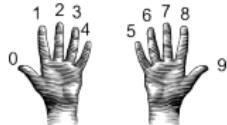
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

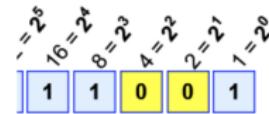
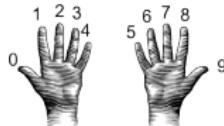
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

3/4 is 0 remainder 3.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

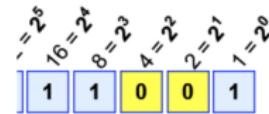
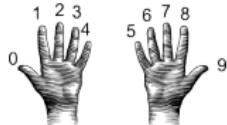
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

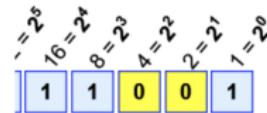
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

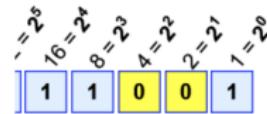
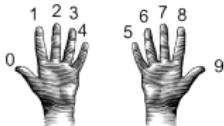
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

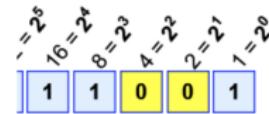
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

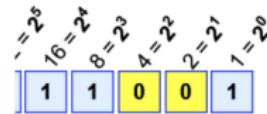
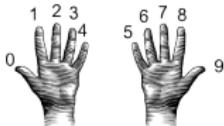
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

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

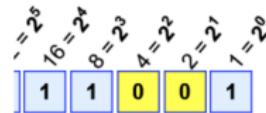
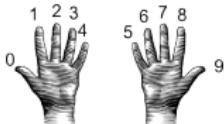
3/8 is 0 rem 3. Next digit is 0: 01100...

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

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

Adding the last remainder: 01100011

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

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

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

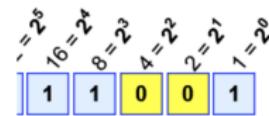
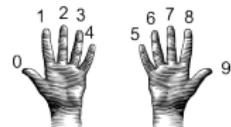
3/4 is 0 remainder 3. Next digit is 0: 011000...

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

Adding the last remainder: 01100011

Answer is 1100011.

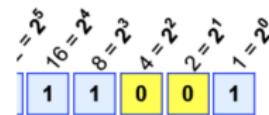
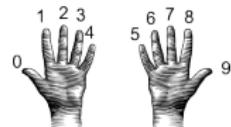
Binary to Decimal: Converting Between Bases



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

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

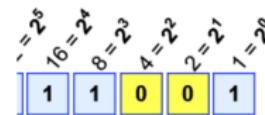
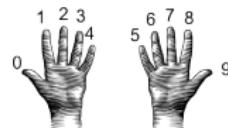
Binary to Decimal: Converting Between Bases



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

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

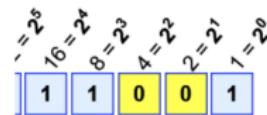
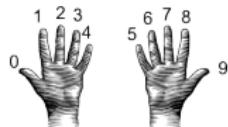
Binary to Decimal: Converting Between Bases



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

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

Binary to Decimal: Converting Between Bases

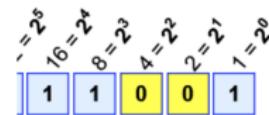
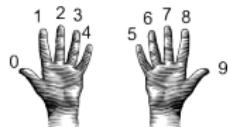


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

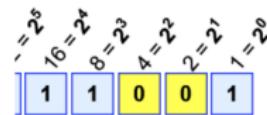
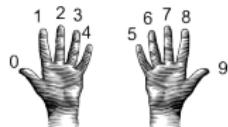


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

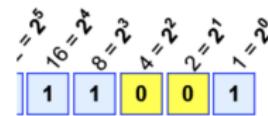
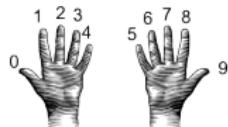


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

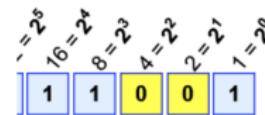
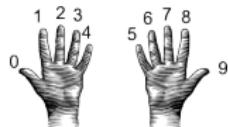


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

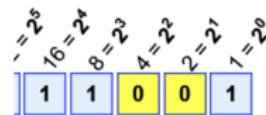
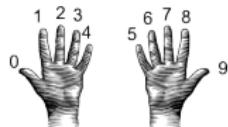


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

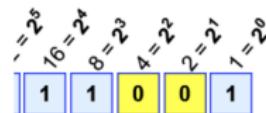


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases



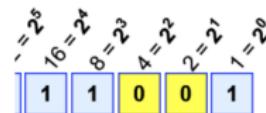
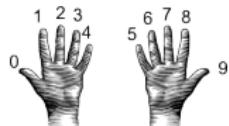
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- From binary to decimal:

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

Sum starts with:

Binary to Decimal: Converting Between Bases



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

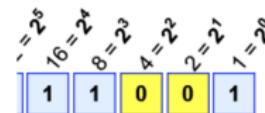
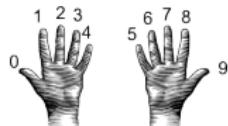
- From binary to decimal:

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

Sum starts with: 1

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

Binary to Decimal: Converting Between Bases



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

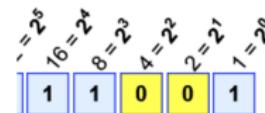
- From binary to decimal:

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

Sum starts with: 1

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

Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

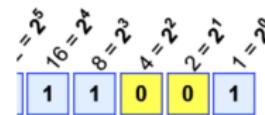
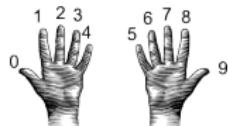
- Set sum = last digit.
- Multiply next digit by 2^1 . Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$. Add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

Sum starts with: 1

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

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

Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

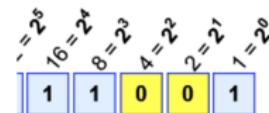
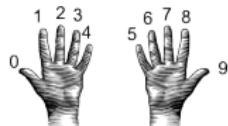
- Set sum = last digit.
- Multiply next digit by $2 = 2^1$. Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$.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

Sum starts with: 1

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

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

Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

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

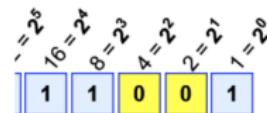
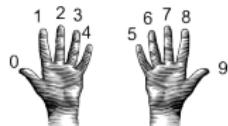
Sum starts with: 1

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

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

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

Binary to Decimal: Converting Between Bases



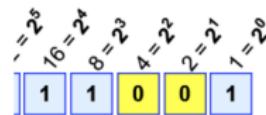
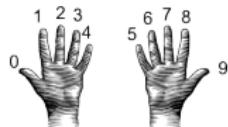
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



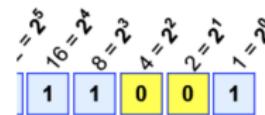
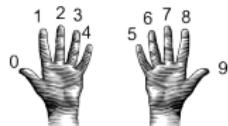
Example: $1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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
 $1 \times 8 = 8$. Add 8 to sum: 13
 $1 \times 16 = 16$. Add 16 to sum:

Binary to Decimal: Converting Between Bases



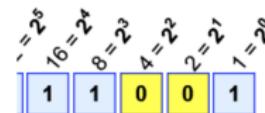
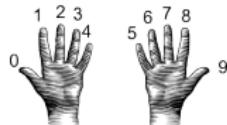
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



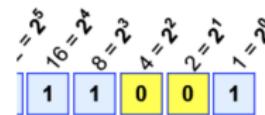
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



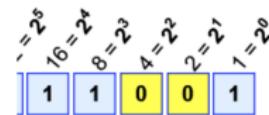
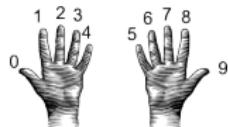
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



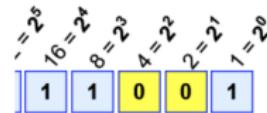
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases

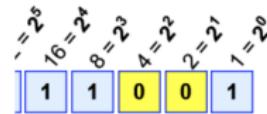
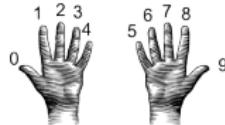


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

Binary to Decimal: Converting Between Bases



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

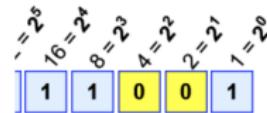
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

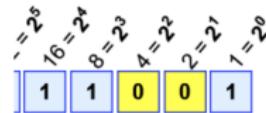
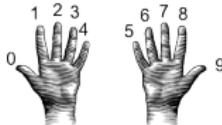
Sum starts with:

0

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

0

Binary to Decimal: Converting Between Bases



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

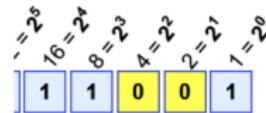
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

Binary to Decimal: Converting Between Bases



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

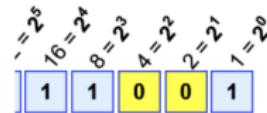
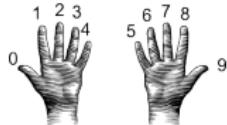
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

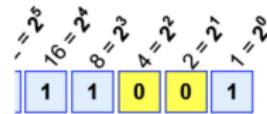
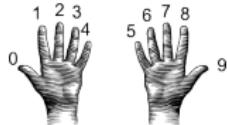
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

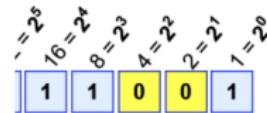
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

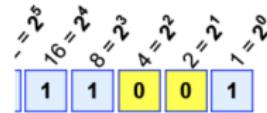
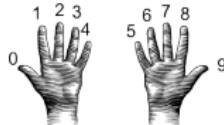
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

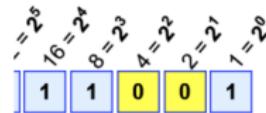
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

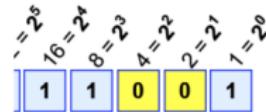
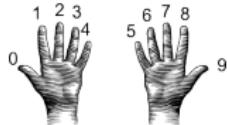
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases

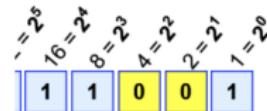


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

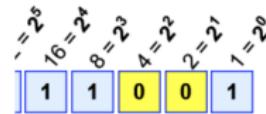
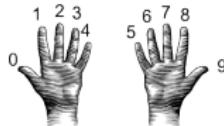
$0 \times 8 = 0$. Add 0 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

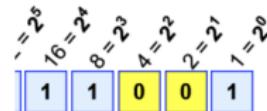
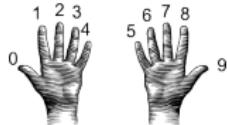
$0 \times 8 = 0$. Add 0 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

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

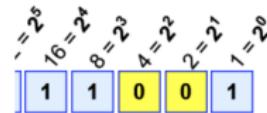
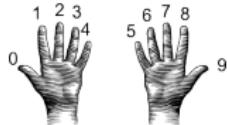
$0 \times 16 = 0$. Add 0 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases

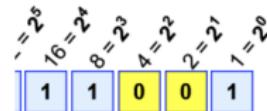
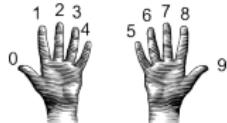


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases



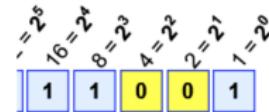
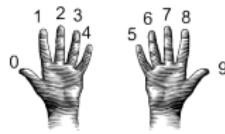
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

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

The answer is 164.

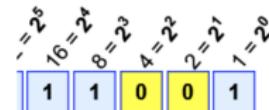
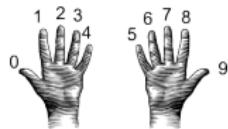
Design Challenge: Incrementers



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

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

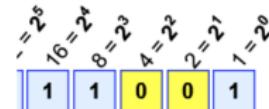
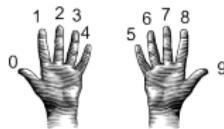
Design Challenge: Incrementers



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

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

Design Challenge: Incrementers

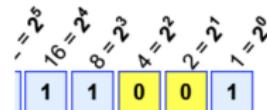
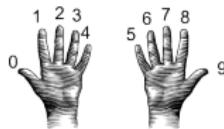


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

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

Design Challenge: Incrementers



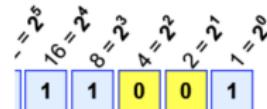
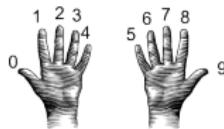
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

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

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

Design Challenge: Incrementers



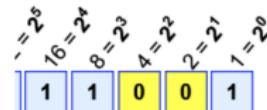
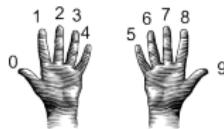
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

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

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

Design Challenge: Incrementers



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

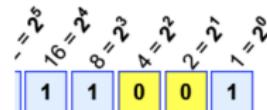
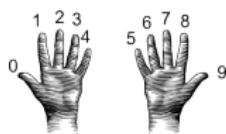
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

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

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

Design Challenge: Incrementers



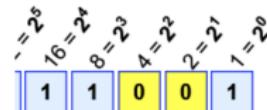
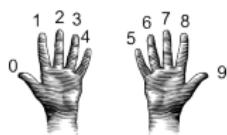
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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 on during the time of the exam

Final Overview: Administration

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

Final Overview: Administration

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

Final Overview: Administration

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

Final Overview: Administration

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

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Final Overview: Format

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

Exam Options

Exam Times:

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

19 December 2008

Exam Rules

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

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

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

Exam Options

Exam Times:

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

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

19 December 2008

Exam Rules

- Show off your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper.
- All work must be handwritten.
- You may take breaks, you may have water, pens and pencils, and your note sheet.
- You are not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

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

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

Exam Options

Exam Times:

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

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

19 December 2008

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper that contains pre-arranged formulas.
- You may take breaks, you may have water, pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

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

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

Exam Options

Exam Times:

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

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

19 December 2020

Exam Rules

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

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

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

Exam Options

Exam Times:

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

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

19 December 2020

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that contains programs, formulas, or other relevant information. This is to prevent CSCI 127 from being accused of academic integrity violations.
- You may bring into the exam room your pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

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

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

Exam Options

Exam Times:

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

Grading Options:

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

19 December 2020

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that contains prewritten formulas and constants. This is the same paper used for CSci 127 on Academic Integrity and will serve as proof of academic dishonesty according to the Hunter College Academic Integrity Policy.
- You may bring a calculator, reference tables, answer sheet, or other electronic device.
- Do not open this exam until instructed to do so.

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

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

Exam Options

Exam Times:

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

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

19 December 2020

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that contains programs and formulas. This is a CUNY-wide rule. No calculators, phones, or other electronic devices.
- You may bring in, you may have in, or you pass, and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

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

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

Grading Options:

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

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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

Weekly Reminders!



Before next lecture, don't forget to:

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