

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

- This lecture will be recorded

Announcements

- Great mentoring opportunity available!
Please check out BB announcement from
yesterday 4/19



Announcements

- Great mentoring opportunity available! Please check out BB announcement from yesterday 4/19
- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**



Announcements



- Great mentoring opportunity available! Please check out BB announcement from yesterday 4/19
- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?

Announcements



- Great mentoring opportunity available! Please check out BB announcement from yesterday 4/19
- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?
 - ▶ Get an A (+)

Announcements



- Great mentoring opportunity available! Please check out BB announcement from yesterday 4/19
- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?
 - ▶ Get an A (+)
 - ▶ Be very comfortable with our labs and programming assignments

Announcements



- Great mentoring opportunity available! Please check out BB announcement from yesterday 4/19
- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?
 - ▶ Get an A (+)
 - ▶ Be very comfortable with our labs and programming assignments
 - ▶ Be able to describe a successful tutoring experience (go to tutoring!!!)

Announcements



- Great mentoring opportunity available! Please check out BB announcement from yesterday 4/19
- I will be recruiting UTAs at the end of this term. You will hear from me by email after the final exam if you earn an A (+), **if you are interested keep an eye out for my emails.**
- What should you do?
 - ▶ Get an A (+)
 - ▶ Be very comfortable with our labs and programming assignments
 - ▶ Be able to describe a successful tutoring experience (go to tutoring!!!)
 - ▶ Previous tutoring experience helpful but not expected

Frequently Asked Questions

From email and tutoring.

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

Frequently Asked Questions

From email and tutoring.

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

The official final is Monday, 24 May, 9-11am.

Frequently Asked Questions

From email and tutoring.

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

The official final is Monday, 24 May, 9-11am.

The early final exam (alternative date) is on Friday, 21 May, 8-10am.

Frequently Asked Questions

From email and tutoring.

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

The official final is Monday, 24 May, 9-11am.

The early final exam (alternative date) is on Friday, 21 May, 8-10am.

Instead of a review sheet, we have:

Frequently Asked Questions

From email and tutoring.

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

The official final is Monday, 24 May, 9-11am.

The early final exam (alternative date) is on Friday, 21 May, 8-10am.

Instead of a review sheet, we have:

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

Frequently Asked Questions

From email and tutoring.

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

The official final is Monday, 24 May, 9-11am.

The early final exam (alternative date) is on Friday, 21 May, 8-10am.

Instead of a review sheet, we have:

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

Frequently Asked Questions

From email and tutoring.

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

The official final is Monday, 24 May, 9-11am.

The early final exam (alternative date) is on Friday, 21 May, 8-10am.

Instead of a review sheet, we have:

- ▶ *All previous final exams (and answer keys) on the website.*
- ▶ *UTAs in drop-in tutoring happy to review concepts and old exam questions.*
- ▶ *There will be opportunity for some practice during our last meeting on 11 May.*

Today's Topics



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

Today's Topics



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

Predict what the code will do:

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

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

Python Tutor

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

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

(Demo with pythonTutor)

Design Pattern: Linear Search

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

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

- Example of **linear search**.

Design Pattern: Linear Search

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

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

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

Design Pattern: Linear Search

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

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

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

Design Pattern: Linear Search

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

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

- Example of **linear search**.
- Start at the beginning of the list.
- Look at each item, one-by-one.
- 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 Python code editor interface. On the left, the code file 'main.py' is open, containing the following Python code:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

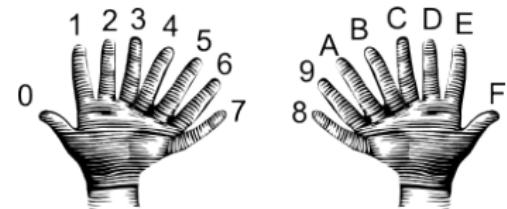
Week 2: variables, data types, more on loops & range()

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

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

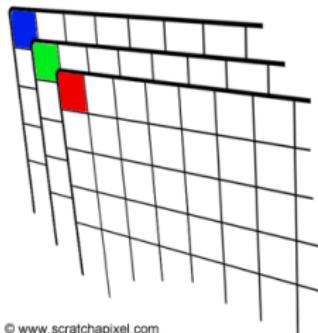
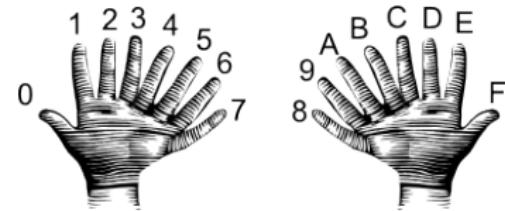
Week 3: colors, hex, slices, numpy & images

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



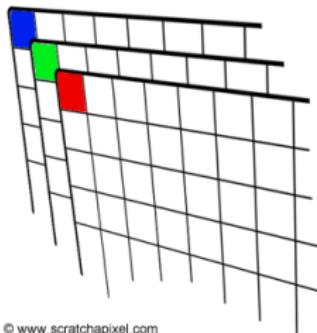
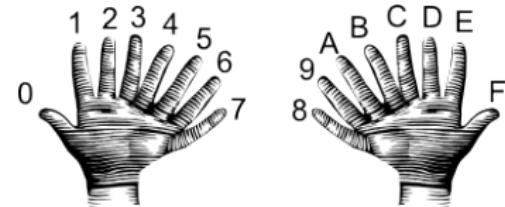
Week 3: colors, hex, slices, numpy & images

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



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

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



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

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

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

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

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

Week 4: design problem (cropping images) & decisions



Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions



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

Week 4: design problem (cropping images) & decisions

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

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

Week 5: logical operators, truth tables & logical circuits

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

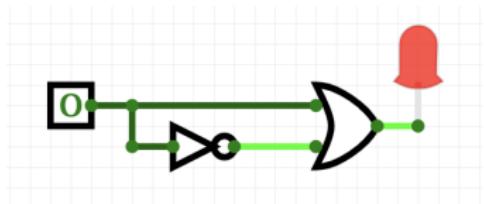
visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
    (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

Week 5: logical operators, truth tables & logical circuits

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

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

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



Week 6: structured data, pandas, & more design

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....
All population figures are consistent with present-day boundaries.....
First census after the consolidation of the five boroughs.....
.....
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1690,1,037,2037,727,788,2842
1771,21843,36232,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,67541,6354,7042,1755,4543,75934
1820,123704,11487,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,120113,140411,5346,10965,391114
1850,355441,128541,189541,5346,10965,391115
1860,613469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51980,33029,1911801
1890,1367711,707128,67243,51980,33029,2141314
1900,185093,116582,152999,200567,67921,2437202
1910,2233142,1634351,284041,430980,8569,476683
1920,2211103,2018354,446031,446031,73201,11651,59148
1930,1867128,152999,152999,152999,152999,4930446
1940,1889924,2469285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7891957
1960,1690101,2738175,1899359,1451277,191555,7891984
1970,1539231,2469285,1471071,1471071,135443,7891984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2485326,2229379,1332450,419782,8080879
2010,1583873,2504705,2229722,1385108,447558,8175133
2015,1444518,2436733,2339150,1459444,474558,8059405

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City,
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
1690,4937,2037,,727,7881
1771,21843,3623,,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,67541,6254,6840,2130,4973,85934
1820,123704,11187,8246,2792,6135,152056
1830,202589,20535,9049,3023,7082,242278
1840,312110,18013,14030,5348,10965,391114
1850,355491,21890,18591,5815,11530,441115
1860,613469,279122,32903,23593,25492,174777
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59943,5653,51980,33091,1911801
1890,1364473,71041,6138,56034,33091,2141134
1900,185093,116582,152999,200567,67921,2437202
1910,2233142,1634351,284041,430980,8569,4766803
1920,2210103,2018354,446071,446071,73201,11651,51048
1930,1867137,1796128,35254,35254,5831,4930446
1940,1889924,2469285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7891957
1960,1690101,2319175,1809049,1451277,191555,7891984
1970,1539231,2460712,1472701,1472701,135443,7891984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,439787,77322564
2000,1537195,2485326,2229379,1332450,439787,8080879
2010,1583873,2504705,2277722,1385108,439787,8175133
2015,1444018,2646733,2339150,1459444,474558,8056405

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

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

```
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,Bronx,Brooklyn,Queens,Bronx,Staten Island,Totals  
1690,4937,2017,...,727,7881  
1771,21843,36232,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75935  
1810,68031,6240,6842,1755,4543,79334  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31150,10,113,1403,5346,10965,391114  
1850,35549,12850,18850,12850,12850,12850  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59945,5653,51980,33029,1911801  
1890,1364473,66531,5653,51980,33029,1911801  
1900,185093,116582,152999,200567,67921,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,22161103,2018354,4460,4460,73201,11651,593083  
1930,16671103,16671103,16671103,16671103,16671103,4930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7891957  
1960,16960101,16960101,16960101,16960101,16960101,781984  
1970,1539331,1465701,1471071,1471071,135443,7808462  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1302789,379977,7322564  
2000,1537195,2485326,2229379,1332650,419782,8080879  
2010,1583873,2504705,2277722,1385108,419782,8175133  
2015,1444518,2540733,2339150,1459446,474558,8059405
```

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,Population  
1690,203,2037,...,727,7181  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,70000,6350,7000,1800,5300,93734  
1820,123704,11187,8246,2792,6135,152056  
1830,20589,20535,9049,3023,7082,242278  
1840,31510,21013,14000,5348,10965,391114  
1850,35549,21800,18500,5800,11000,45115  
1860,813469,279122,23903,23933,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1367000,71000,65000,51000,35000,210000  
1900,1850093,116582,152999,200567,67621,2437202  
1910,2331842,1634351,2841,430980,8569,476683  
1920,2246103,2018354,44601,72021,11651,50048  
1930,1867112,1578128,1578128,1578128,1578128,4930446  
1940,1889924,2469285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690000,2000000,1800000,1600000,1400000,781984  
1970,1539231,1465070,1472701,1472701,135443,769462  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,737987,77322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,4175133,8175133  
2015,1444518,2436733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6

Week 6: structured data, pandas, & more design

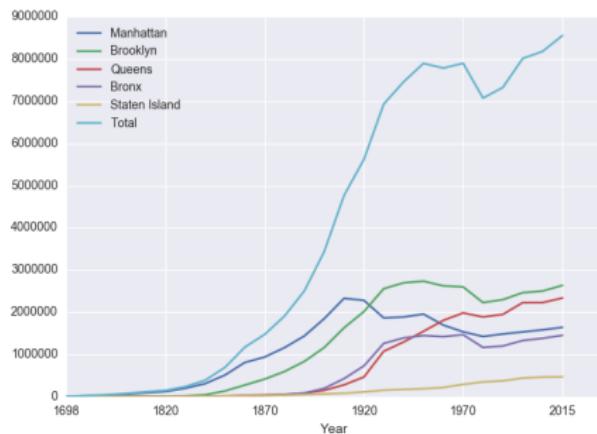
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
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  
1698,Manhattan,2037,727,7181  
1771,21843,36231,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,71531,6349,7041,1813,5197,93734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14031,5348,10965,391114  
1850,355441,21803,18951,5851,12001,45115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33051,1911801  
1890,1384473,72001,6348,57348,38501,1911814  
1900,1850993,1165852,152999,200567,67921,2437202  
1910,2331842,1634351,2841,430980,8569,476683  
1920,2281103,2018354,44601,732013,11651,500488  
1930,2667103,2480128,479128,52545,5831,630446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7891957  
1960,1696101,2738175,1550849,1451277,191555,7891984  
1970,1539231,2465701,1471701,1271701,135443,7891984  
1980,1426285,2230936,11891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332450,413728,8080879  
2010,1583873,2504705,2216722,1385108,413728,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6



Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 7: functions

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

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

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

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

Week 8: function parameters, github

- Functions can have **input parameters**.

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

Week 8: function parameters, github

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

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

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

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

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



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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

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

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

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

```
import turtle
import random

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

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

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

Python & Circuits Review: 10 Weeks in 10 Minutes



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

Lecture Quiz

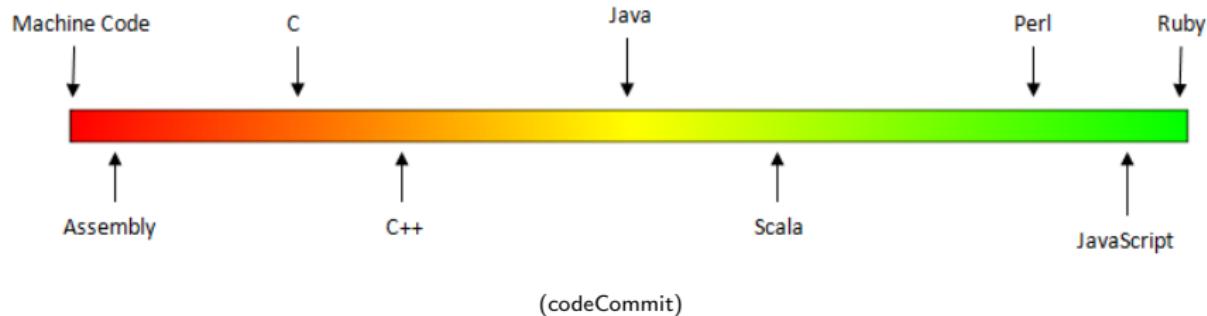
- Log-in to Gradescope
- Find LECTURE 11 Quiz
- Take the quiz
- **You have 3 minutes**

Today's Topics



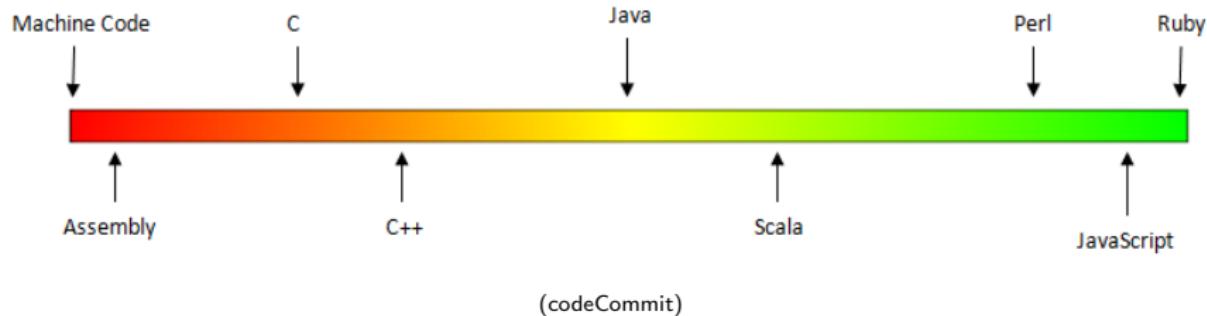
- Design Patterns: Searching
- Python Recap
- **Machine Language**
- Machine Language: Jumps & Loops
- Binary & Hex Arithmetic
- Final Exam: Format

Low-Level vs. High-Level Languages



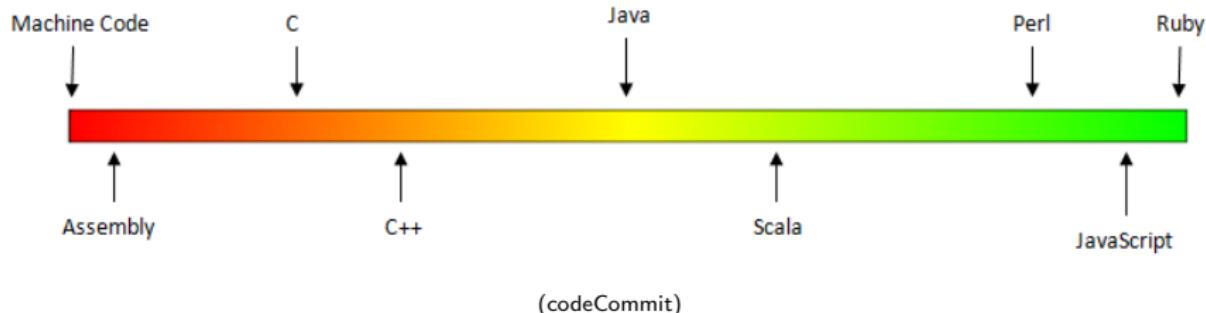
- Can view programming languages on a continuum.

Low-Level vs. High-Level Languages



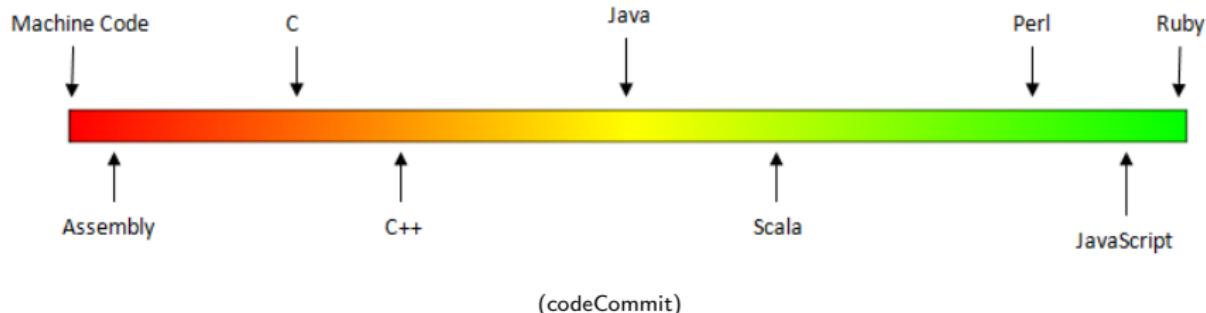
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages**

Low-Level vs. High-Level Languages



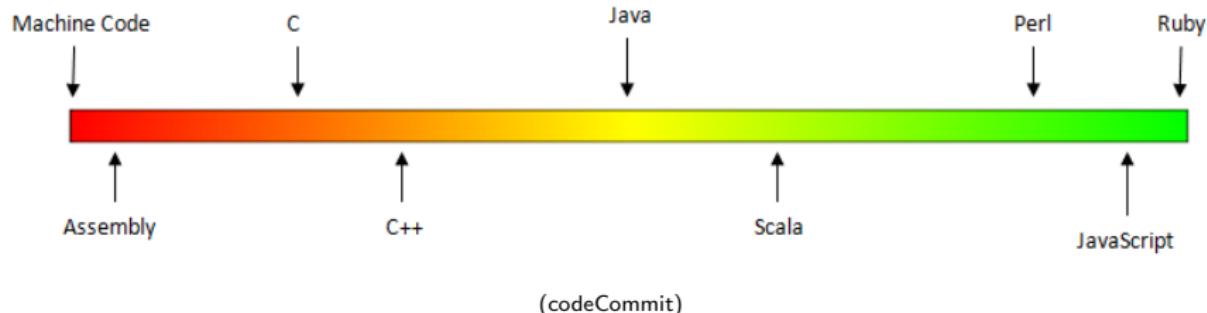
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).

Low-Level vs. High-Level Languages



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

Low-Level vs. High-Level Languages



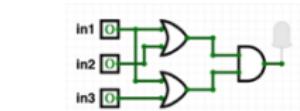
- Can view programming languages on a continuum.
- Those that directly access machine instructions & memory and have little abstraction are **low-level languages** (e.g. machine language, assembly language).
- Those that have strong abstraction (allow programming paradigms independent of the machine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

Processing

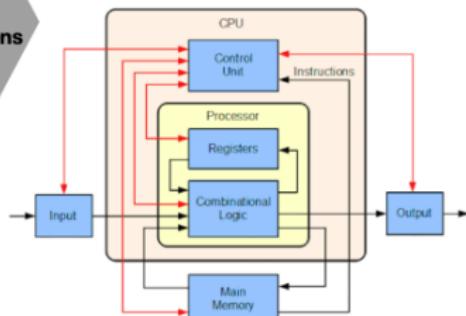
Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.



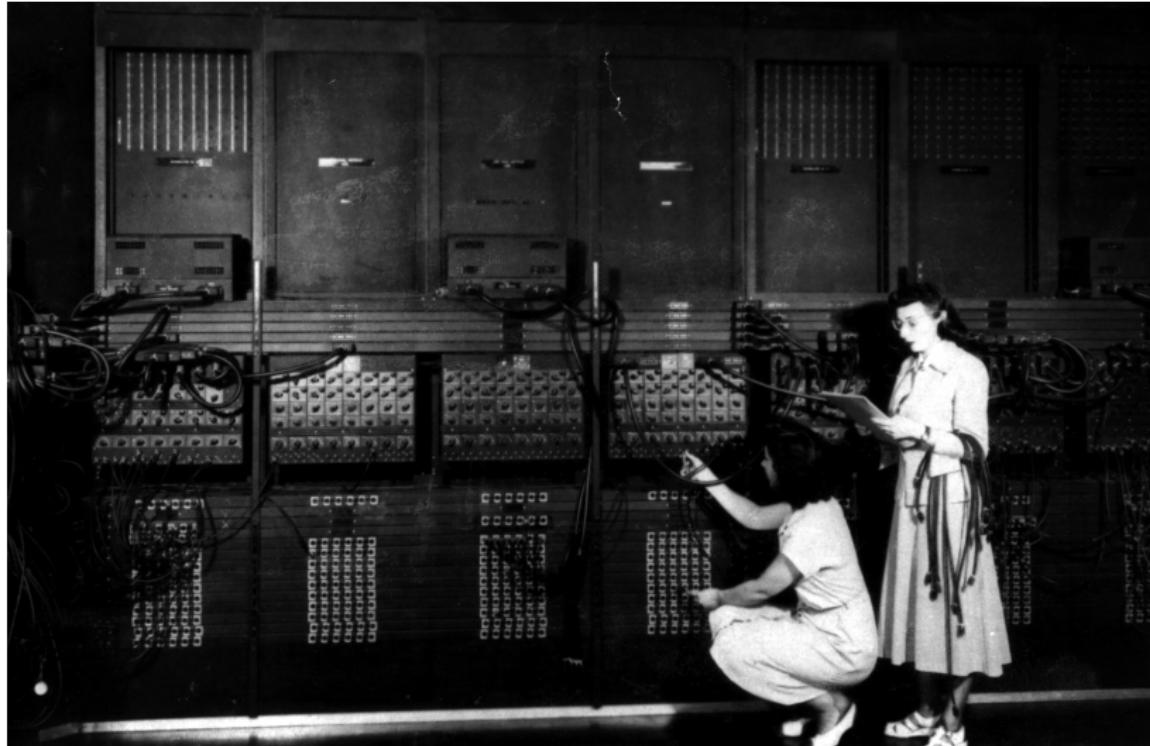
```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)
```



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



Machine Language



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

Machine Language

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

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

BREAK

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

(wiki)

Machine Language

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

(wiki)

Machine Language

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

(wiki)

Machine Language



The screenshot shows a terminal window with assembly code and its corresponding binary output. The assembly code includes instructions like LDI, GCD, LDH, ADC, STA, CLD, SED, and SWI. The binary output consists of two columns of hex values.

Assembly Instruction	Binary Value
LDI \$4321, \$0000	00 0000 C2 30
GCD	00 0000 FB
LDH \$0, \$4321	00 0003 00 00
ADC \$0, \$4321	00 0007 00 00
STA \$0, \$4321	00 000F 00 00
CLD	00 0011 00 00
SED	00 0011 00 00
SWI	00 0011 00 00

(wiki)

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

Machine Language

(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
```

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

(WeMIPS)

WeMIPS

The screenshot shows the WeMIPS debugger interface. At the top, there are tabs for 'Live', '3', 'Data', and 'ShowHide Device'. Below these are navigation links: 'Addition Doubler', 'Stax', 'Looper', 'Stack Test', and 'Hello World'. Underneath are buttons for 'Code Gen Save String', 'Interactive', 'Binary2 Decimal', and 'Decimal2 Binary'. A 'Debug' button is also present.

The main area contains assembly code:

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

MIPS Commands

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

MIPS Commands

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

MIPS Commands

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

MIPS Commands

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

MIPS Commands

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

MIPS Commands

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

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

MIPS Commands

The screenshot shows a MIPS assembly debugger interface. 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 watching', and 'Stop' buttons. A status bar at the bottom right says 'User Guide | Unit Tests | Docs'.

The main area contains assembly code:

```
1 # Shows 'Hello world' at the top of the stack
2 .text
3 .globl _start
4 .type _start, @function
5 _start:
6    li $t0, 111901
7    li $t1, 111902
8    li $t2, 111903
9    li $t3, 111904
10   li $t4, 111905
11   li $t5, 111906
12   li $t6, 111907
13   li $t7, 111908
14   li $t8, 111909
15   li $t9, 11190a
16   li $t10, 11190b
17   li $t11, 11190c
18   li $t12, 11190d
19   li $t13, 11190e
20   li $t14, 11190f
21   li $t15, 111910
22   li $t16, 111911
23   li $t17, 111912
24   li $t18, 111913
25   li $t19, 111914
26   li $t20, 111915
27   li $t21, 111916
28   li $t22, 111917
29   li $t23, 111918
30   li $t24, 111919
31   li $t25, 11191a
32   li $t26, 11191b
33   li $t27, 11191c
34   li $t28, 11191d
35   li $t29, 11191e
36   li $t30, 11191f
37   li $t31, 111920
38   addi $t0, $t0, 128
39   addi $t1, $t1, 128
40   addi $t2, $t2, 128
41   addi $t3, $t3, 128
42   addi $t4, $t4, 128
43   addi $t5, $t5, 128
44   addi $t6, $t6, 128
45   addi $t7, $t7, 128
46   addi $t8, $t8, 128
47   addi $t9, $t9, 128
48   addi $t10, $t10, 128
49   addi $t11, $t11, 128
50   addi $t12, $t12, 128
51   addi $t13, $t13, 128
52   addi $t14, $t14, 128
53   addi $t15, $t15, 128
54   addi $t16, $t16, 128
55   addi $t17, $t17, 128
56   addi $t18, $t18, 128
57   addi $t19, $t19, 128
58   addi $t20, $t20, 128
59   addi $t21, $t21, 128
60   addi $t22, $t22, 128
61   addi $t23, $t23, 128
62   addi $t24, $t24, 128
63   addi $t25, $t25, 128
64   addi $t26, $t26, 128
65   addi $t27, $t27, 128
66   addi $t28, $t28, 128
67   addi $t29, $t29, 128
68   addi $t30, $t30, 128
69   addi $t31, $t31, 128
70   addi $t0, $t0, 128
71   addi $t1, $t1, 128
72   addi $t2, $t2, 128
73   addi $t3, $t3, 128
74   addi $t4, $t4, 128
75   addi $t5, $t5, 128
76   addi $t6, $t6, 128
77   addi $t7, $t7, 128
78   addi $t8, $t8, 128
79   addi $t9, $t9, 128
80   addi $t10, $t10, 128
81   addi $t11, $t11, 128
82   addi $t12, $t12, 128
83   addi $t13, $t13, 128
84   addi $t14, $t14, 128
85   addi $t15, $t15, 128
86   addi $t16, $t16, 128
87   addi $t17, $t17, 128
88   addi $t18, $t18, 128
89   addi $t19, $t19, 128
90   addi $t20, $t20, 128
91   addi $t21, $t21, 128
92   addi $t22, $t22, 128
93   addi $t23, $t23, 128
94   addi $t24, $t24, 128
95   addi $t25, $t25, 128
96   addi $t26, $t26, 128
97   addi $t27, $t27, 128
98   addi $t28, $t28, 128
99   addi $t29, $t29, 128
100  addi $t30, $t30, 128
101  addi $t31, $t31, 128
102  addi $t0, $t0, 128
103  addi $t1, $t1, 128
104  addi $t2, $t2, 128
105  addi $t3, $t3, 128
106  addi $t4, $t4, 128
107  addi $t5, $t5, 128
108  addi $t6, $t6, 128
109  addi $t7, $t7, 128
110  addi $t8, $t8, 128
111  addi $t9, $t9, 128
112  addi $t10, $t10, 128
113  addi $t11, $t11, 128
114  addi $t12, $t12, 128
115  addi $t13, $t13, 128
116  addi $t14, $t14, 128
117  addi $t15, $t15, 128
118  addi $t16, $t16, 128
119  addi $t17, $t17, 128
120  addi $t18, $t18, 128
121  addi $t19, $t19, 128
122  addi $t20, $t20, 128
123  addi $t21, $t21, 128
124  addi $t22, $t22, 128
125  addi $t23, $t23, 128
126  addi $t24, $t24, 128
127  addi $t25, $t25, 128
128  addi $t26, $t26, 128
129  addi $t27, $t27, 128
130  addi $t28, $t28, 128
131  addi $t29, $t29, 128
132  addi $t30, $t30, 128
133  addi $t31, $t31, 128
134  addi $t0, $t0, 128
135  addi $t1, $t1, 128
136  addi $t2, $t2, 128
137  addi $t3, $t3, 128
138  addi $t4, $t4, 128
139  addi $t5, $t5, 128
140  addi $t6, $t6, 128
141  addi $t7, $t7, 128
142  addi $t8, $t8, 128
143  addi $t9, $t9, 128
144  addi $t10, $t10, 128
145  addi $t11, $t11, 128
146  addi $t12, $t12, 128
147  addi $t13, $t13, 128
148  addi $t14, $t14, 128
149  addi $t15, $t15, 128
150  addi $t16, $t16, 128
151  addi $t17, $t17, 128
152  addi $t18, $t18, 128
153  addi $t19, $t19, 128
154  addi $t20, $t20, 128
155  addi $t21, $t21, 128
156  addi $t22, $t22, 128
157  addi $t23, $t23, 128
158  addi $t24, $t24, 128
159  addi $t25, $t25, 128
160  addi $t26, $t26, 128
161  addi $t27, $t27, 128
162  addi $t28, $t28, 128
163  addi $t29, $t29, 128
164  addi $t30, $t30, 128
165  addi $t31, $t31, 128
166  addi $t0, $t0, 128
167  addi $t1, $t1, 128
168  addi $t2, $t2, 128
169  addi $t3, $t3, 128
170  addi $t4, $t4, 128
171  addi $t5, $t5, 128
172  addi $t6, $t6, 128
173  addi $t7, $t7, 128
174  addi $t8, $t8, 128
175  addi $t9, $t9, 128
176  addi $t10, $t10, 128
177  addi $t11, $t11, 128
178  addi $t12, $t12, 128
179  addi $t13, $t13, 128
180  addi $t14, $t14, 128
181  addi $t15, $t15, 128
182  addi $t16, $t16, 128
183  addi $t17, $t17, 128
184  addi $t18, $t18, 128
185  addi $t19, $t19, 128
186  addi $t20, $t20, 128
187  addi $t21, $t21, 128
188  addi $t22, $t22, 128
189  addi $t23, $t23, 128
190  addi $t24, $t24, 128
191  addi $t25, $t25, 128
192  addi $t26, $t26, 128
193  addi $t27, $t27, 128
194  addi $t28, $t28, 128
195  addi $t29, $t29, 128
196  addi $t30, $t30, 128
197  addi $t31, $t31, 128
198  addi $t0, $t0, 128
199  addi $t1, $t1, 128
200  addi $t2, $t2, 128
201  addi $t3, $t3, 128
202  addi $t4, $t4, 128
203  addi $t5, $t5, 128
204  addi $t6, $t6, 128
205  addi $t7, $t7, 128
206  addi $t8, $t8, 128
207  addi $t9, $t9, 128
208  addi $t10, $t10, 128
209  addi $t11, $t11, 128
210  addi $t12, $t12, 128
211  addi $t13, $t13, 128
212  addi $t14, $t14, 128
213  addi $t15, $t15, 128
214  addi $t16, $t16, 128
215  addi $t17, $t17, 128
216  addi $t18, $t18, 128
217  addi $t19, $t19, 128
218  addi $t20, $t20, 128
219  addi $t21, $t21, 128
220  addi $t22, $t22, 128
221  addi $t23, $t23, 128
222  addi $t24, $t24, 128
223  addi $t25, $t25, 128
224  addi $t26, $t26, 128
225  addi $t27, $t27, 128
226  addi $t28, $t28, 128
227  addi $t29, $t29, 128
228  addi $t30, $t30, 128
229  addi $t31, $t31, 128
230  addi $t0, $t0, 128
231  addi $t1, $t1, 128
232  addi $t2, $t2, 128
233  addi $t3, $t3, 128
234  addi $t4, $t4, 128
235  addi $t5, $t5, 128
236  addi $t6, $t6, 128
237  addi $t7, $t7, 128
238  addi $t8, $t8, 128
239  addi $t9, $t9, 128
240  addi $t10, $t10, 128
241  addi $t11, $t11, 128
242  addi $t12, $t12, 128
243  addi $t13, $t13, 128
244  addi $t14, $t14, 128
245  addi $t15, $t15, 128
246  addi $t16, $t16, 128
247  addi $t17, $t17, 128
248  addi $t18, $t18, 128
249  addi $t19, $t19, 128
250  addi $t20, $t20, 128
251  addi $t21, $t21, 128
252  addi $t22, $t22, 128
253  addi $t23, $t23, 128
254  addi $t24, $t24, 128
255  addi $t25, $t25, 128
256  addi $t26, $t26, 128
257  addi $t27, $t27, 128
258  addi $t28, $t28, 128
259  addi $t29, $t29, 128
260  addi $t30, $t30, 128
261  addi $t31, $t31, 128
262  addi $t0, $t0, 128
263  addi $t1, $t1, 128
264  addi $t2, $t2, 128
265  addi $t3, $t3, 128
266  addi $t4, $t4, 128
267  addi $t5, $t5, 128
268  addi $t6, $t6, 128
269  addi $t7, $t7, 128
270  addi $t8, $t8, 128
271  addi $t9, $t9, 128
272  addi $t10, $t10, 128
273  addi $t11, $t11, 128
274  addi $t12, $t12, 128
275  addi $t13, $t13, 128
276  addi $t14, $t14, 128
277  addi $t15, $t15, 128
278  addi $t16, $t16, 128
279  addi $t17, $t17, 128
280  addi $t18, $t18, 128
281  addi $t19, $t19, 128
282  addi $t20, $t20, 128
283  addi $t21, $t21, 128
284  addi $t22, $t22, 128
285  addi $t23, $t23, 128
286  addi $t24, $t24, 128
287  addi $t25, $t25, 128
288  addi $t26, $t26, 128
289  addi $t27, $t27, 128
290  addi $t28, $t28, 128
291  addi $t29, $t29, 128
292  addi $t30, $t30, 128
293  addi $t31, $t31, 128
294  addi $t0, $t0, 128
295  addi $t1, $t1, 128
296  addi $t2, $t2, 128
297  addi $t3, $t3, 128
298  addi $t4, $t4, 128
299  addi $t5, $t5, 128
300  addi $t6, $t6, 128
301  addi $t7, $t7, 128
302  addi $t8, $t8, 128
303  addi $t9, $t9, 128
304  addi $t10, $t10, 128
305  addi $t11, $t11, 128
306  addi $t12, $t12, 128
307  addi $t13, $t13, 128
308  addi $t14, $t14, 128
309  addi $t15, $t15, 128
310  addi $t16, $t16, 128
311  addi $t17, $t17, 128
312  addi $t18, $t18, 128
313  addi $t19, $t19, 128
314  addi $t20, $t20, 128
315  addi $t21, $t21, 128
316  addi $t22, $t22, 128
317  addi $t23, $t23, 128
318  addi $t24, $t24, 128
319  addi $t25, $t25, 128
320  addi $t26, $t26, 128
321  addi $t27, $t27, 128
322  addi $t28, $t28, 128
323  addi $t29, $t29, 128
324  addi $t30, $t30, 128
325  addi $t31, $t31, 128
326  addi $t0, $t0, 128
327  addi $t1, $t1, 128
328  addi $t2, $t2, 128
329  addi $t3, $t3, 128
330  addi $t4, $t4, 128
331  addi $t5, $t5, 128
332  addi $t6, $t6, 128
333  addi $t7, $t7, 128
334  addi $t8, $t8, 128
335  addi $t9, $t9, 128
336  addi $t10, $t10, 128
337  addi $t11, $t11, 128
338  addi $t12, $t12, 128
339  addi $t13, $t13, 128
340  addi $t14, $t14, 128
341  addi $t15, $t15, 128
342  addi $t16, $t16, 128
343  addi $t17, $t17, 128
344  addi $t18, $t18, 128
345  addi $t19, $t19, 128
346  addi $t20, $t20, 128
347  addi $t21, $t21, 128
348  addi $t22, $t22, 128
349  addi $t23, $t23, 128
350  addi $t24, $t24, 128
351  addi $t25, $t25, 128
352  addi $t26, $t26, 128
353  addi $t27, $t27, 128
354  addi $t28, $t28, 128
355  addi $t29, $t29, 128
356  addi $t30, $t30, 128
357  addi $t31, $t31, 128
358  addi $t0, $t0, 128
359  addi $t1, $t1, 128
360  addi $t2, $t2, 128
361  addi $t3, $t3, 128
362  addi $t4, $t4, 128
363  addi $t5, $t5, 128
364  addi $t6, $t6, 128
365  addi $t7, $t7, 128
366  addi $t8, $t8, 128
367  addi $t9, $t9, 128
368  addi $t10, $t10, 128
369  addi $t11, $t11, 128
370  addi $t12, $t12, 128
371  addi $t13, $t13, 128
372  addi $t14, $t14, 128
373  addi $t15, $t15, 128
374  addi $t16, $t16, 128
375  addi $t17, $t17, 128
376  addi $t18, $t18, 128
377  addi $t19, $t19, 128
378  addi $t20, $t20, 128
379  addi $t21, $t21, 128
380  addi $t22, $t22, 128
381  addi $t23, $t23, 128
382  addi $t24, $t24, 128
383  addi $t25, $t25, 128
384  addi $t26, $t26, 128
385  addi $t27, $t27, 128
386  addi $t28, $t28, 128
387  addi $t29, $t29, 128
388  addi $t30, $t30, 128
389  addi $t31, $t31, 128
390  addi $t0, $t0, 128
391  addi $t1, $t1, 128
392  addi $t2, $t2, 128
393  addi $t3, $t3, 128
394  addi $t4, $t4, 128
395  addi $t5, $t5, 128
396  addi $t6, $t6, 128
397  addi $t7, $t7, 128
398  addi $t8, $t8, 128
399  addi $t9, $t9, 128
400  addi $t10, $t10, 128
401  addi $t11, $t11, 128
402  addi $t12, $t12, 128
403  addi $t13, $t13, 128
404  addi $t14, $t14, 128
405  addi $t15, $t15, 128
406  addi $t16, $t16, 128
407  addi $t17, $t17, 128
408  addi $t18, $t18, 128
409  addi $t19, $t19, 128
410  addi $t20, $t20, 128
411  addi $t21, $t21, 128
412  addi $t22, $t22, 128
413  addi $t23, $t23, 128
414  addi $t24, $t24, 128
415  addi $t25, $t25, 128
416  addi $t26, $t26, 128
417  addi $t27, $t27, 128
418  addi $t28, $t28, 128
419  addi $t29, $t29, 128
420  addi $t30, $t30, 128
421  addi $t31, $t31, 128
422  addi $t0, $t0, 128
423  addi $t1, $t1, 128
424  addi $t2, $t2, 128
425  addi $t3, $t3, 128
426  addi $t4, $t4, 128
427  addi $t5, $t5, 128
428  addi $t6, $t6, 128
429  addi $t7, $t7, 128
430  addi $t8, $t8, 128
431  addi $t9, $t9, 128
432  addi $t10, $t10, 128
433  addi $t11, $t11, 128
434  addi $t12, $t12, 128
435  addi $t13, $t13, 128
436  addi $t14, $t14, 128
437  addi $t15, $t15, 128
438  addi $t16, $t16, 128
439  addi $t17, $t17, 128
440  addi $t18, $t18, 128
441  addi $t19, $t19, 128
442  addi $t20, $t20, 128
443  addi $t21, $t21, 128
444  addi $t22, $t22, 128
445  addi $t23, $t23, 128
446  addi $t24, $t24, 128
447  addi $t25, $t25, 128
448  addi $t26, $t26, 128
449  addi $t27, $t27, 128
450  addi $t28, $t28, 128
451  addi $t29, $t29, 128
452  addi $t30, $t30, 128
453  addi $t31, $t31, 128
454  addi $t0, $t0, 128
455  addi $t1, $t1, 128
456  addi $t2, $t2, 128
457  addi $t3, $t3, 128
458  addi $t4, $t4, 128
459  addi $t5, $t5, 128
460  addi $t6, $t6, 128
461  addi $t7, $t7, 128
462  addi $t8, $t8, 128
463  addi $t9, $t9, 128
464  addi $t10, $t10, 128
465  addi $t11, $t11, 128
466  addi $t12, $t12, 128
467  addi $t13, $t13, 128
468  addi $t14, $t14, 128
469  addi $t15, $t15, 128
470  addi $t16, $t16, 128
471  addi $t17, $t17, 128
472  addi $t18, $t18, 128
473  addi $t19, $t19, 128
474  addi $t20, $t20, 128
475  addi $t21, $t21, 128
476  addi $t22, $t22, 128
477  addi $t23, $t23, 128
478  addi $t24, $t24, 128
479  addi $t25, $t25, 128
480  addi $t26, $t26, 128
481  addi $t27, $t27, 128
482  addi $t28, $t28, 128
483  addi $t29, $t29, 128
484  addi $t30, $t30, 128
485  addi $t31, $t31, 128
486  addi $t0, $t0, 128
487  addi $t1, $t1, 128
488  addi $t2, $t2, 128
489  addi $t3, $t3, 128
490  addi $t4, $t4, 128
491  addi $t5, $t5, 128
492  addi $t6, $t6, 128
493  addi $t7, $t7, 128
494  addi $t8, $t8, 128
495  addi $t9, $t9, 128
496  addi $t10, $t10, 128
497  addi $t11, $t11, 128
498  addi $t12, $t12, 128
499  addi $t13, $t13, 128
500  addi $t14, $t14, 128
501  addi $t15, $t15, 128
502  addi $t16, $t16, 128
503  addi $t17, $t17, 128
504  addi $t18, $t18, 128
505  addi $t19, $t19, 128
506  addi $t20, $t20, 128
507  addi $t21, $t21, 128
508  addi $t22, $t22, 128
509  addi $t23, $t23, 128
510  addi $t24, $t24, 128
511  addi $t25, $t25, 128
512  addi $t26, $t26, 128
513  addi $t27, $t27, 128
514  addi $t28, $t28, 128
515  addi $t29, $t29, 128
516  addi $t30, $t30, 128
517  addi $t31, $t31, 128
518  addi $t0, $t0, 128
519  addi $t1, $t1, 128
520  addi $t2, $t2, 128
521  addi $t3, $t3, 128
522  addi $t4, $t4, 128
523  addi $t5, $t5, 128
524  addi $t6, $t6, 128
525  addi $t7, $t7, 128
526  addi $t8, $t8, 128
527  addi $t9, $t9, 128
528  addi $t10, $t10, 128
529  addi $t11, $t11, 128
530  addi $t12, $t12, 128
531  addi $t13, $t13, 128
532  addi $t14, $t14, 128
533  addi $t15, $t15, 128
534  addi $t16, $t16, 128
535  addi $t17, $t17, 128
536  addi $t18, $t18, 128
537  addi $t19, $t19, 128
538  addi $t20, $t20, 128
539  addi $t21, $t21, 128
540  addi $t22, $t22, 128
541  addi $t23, $t23, 128
542  addi $t24, $t24, 128
543  addi $t25, $t25, 128
544  addi $t26, $t26, 128
545  addi $t27, $t27, 128
546  addi $t28, $t28, 128
547  addi $t29, $t29, 128
548  addi $t30, $t30, 128
549  addi $t31, $t31, 128
550  addi $t0, $t0, 128
551  addi $t1, $t1, 128
552  addi $t2, $t2, 128
553  addi $t3, $t3, 128
554  addi $t4, $t4, 128
555  addi $t5, $t5, 128
556  addi $t6, $t6, 128
557  addi $t7, $t7, 128
558  addi $t8, $t8, 128
559  addi $t9, $t9, 128
560  addi $t10, $t10, 128
561  addi $t11, $t11, 128
562  addi $t12, $t12, 128
563  addi $t13, $t13, 128
564  addi $t14, $t14, 128
565  addi $t15, $t15, 128
566  addi $t16, $t16, 128
567  addi $t17, $t17, 128
568  addi $t18, $t18, 128
569  addi $t19, $t19, 128
570  addi $t20, $t20, 128
571  addi $t21, $t21, 128
572  addi $t22, $t22, 128
573  addi $t23, $t23, 128
574  addi $t24, $t24, 128
575  addi $t25, $t25, 128
576  addi $t26, $t26, 128
577  addi $t27, $t27, 128
578  addi $t28, $t28, 128
579  addi $t29, $t29, 128
580  addi $t30, $t30, 128
581  addi $t31, $t31, 128
582  addi $t0, $t0, 128
583  addi $t1, $t1, 128
584  addi $t2, $t2, 128
585  addi $t3, $t3, 128
586  addi $t4, $t4, 128
587  addi $t5, $t5, 128
588  addi $t6, $t6, 128
589  addi $t7, $t7, 128
590  addi $t8, $t8, 128
591  addi $t9, $t9, 128
592  addi $t10, $t10, 128
593  addi $t11, $t11, 128
594  addi $t12, $t12, 128
595  addi $t13, $t13, 128
596  addi $t14, $t14, 128
597  addi $t15, $t15, 128
598  addi $t16, $t16, 128
599  addi $t17, $t17, 128
600  addi $t18, $t18, 128
601  addi $t19, $t19, 128
602  addi $t20, $t20, 128
603  addi $t21, $t21, 128
604  addi $t22, $t22, 128
605  addi $t23, $t23, 128
606  addi $t24, $t24, 128
607  addi $t25, $t25, 128
608  addi $t26, $t26, 128
609  addi $t27, $t27, 128
610  addi $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

d0 10
d1 9
d2 8
d3 20
d4 656
d5 676
d6 807
d7 418

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
ADDQ $10, %esp;    # esp = 0 for prior saving
movl %esp, %ebp;   # points to the top
syscall;
```

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

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. On the right, there's a register table with columns for \$, T, A, V, Stack, and Log. The registers shown are \$0 through \$31, with their values: \$0=10, \$1=9, \$2=8, \$3=22, \$4=000, \$5=819, \$6=827, and \$7=411. The main area contains assembly code:

```
# Shows "Hello world" at the top of the stack
1  li    $t0, 11110000
2  sb    $t0, 8($sp)
3  addi $t0, $t0, 1111 # zero extend
4  sb    $t0, 11110000
5  li    $t1, 11110001
6  sb    $t1, 11110001
7  li    $t2, 11110002
8  sb    $t2, 11110002
9  li    $t3, 11110003
10  sb   $t3, 11110003
11  li    $t4, 11110004
12  sb   $t4, 11110004
13  li    $t5, 11110005
14  sb   $t5, 11110005
15  li    $t6, 11110006
16  sb   $t6, 11110006
17  li    $t7, 11110007
18  sb   $t7, 11110007
19  addi $t8, $t0, $t0
20  addi $t9, $t1, $t1
21  addi $t10, $t2, $t2
22  addi $t11, $t3, $t3
23  addi $t12, $t4, $t4
24  addi $t13, $t5, $t5
25  addi $t14, $t6, $t6
26  addi $t15, $t7, $t7
27  addi $t16, $t8, $t8
28  addi $t17, $t9, $t9
29  addi $t18, $t10, $t10
30  addi $t19, $t11, $t11
31  addi $t20, $t12, $t12
32  addi $t21, $t13, $t13
33  addi $t22, $t14, $t14
34  addi $t23, $t15, $t15
35  addi $t24, $t16, $t16
36  addi $t25, $t17, $t17
37  addi $t26, $t18, $t18
38  addi $t27, $t19, $t19
39  addi $t28, $t20, $t20
40  addi $t29, $t21, $t21
41  addi $t30, $t22, $t22
42  addi $t31, $t23, $t23
43  addi $t32, $t24, $t24
44  addi $t33, $t25, $t25
45  addi $t34, $t26, $t26
46  addi $t35, $t27, $t27
47  addi $t36, $t28, $t28
48  addi $t37, $t29, $t29
49  addi $t38, $t30, $t30
50  addi $t39, $t31, $t31
51  addi $t40, $t32, $t32
52  addi $t41, $t33, $t33
53  addi $t42, $t34, $t34
54  addi $t43, $t35, $t35
55  addi $t44, $t36, $t36
56  addi $t45, $t37, $t37
57  addi $t46, $t38, $t38
58  addi $t47, $t39, $t39
59  addi $t48, $t40, $t40
60  addi $t49, $t41, $t41
61  addi $t50, $t42, $t42
62  addi $t51, $t43, $t43
63  addi $t52, $t44, $t44
64  addi $t53, $t45, $t45
65  addi $t54, $t46, $t46
66  addi $t55, $t47, $t47
67  addi $t56, $t48, $t48
68  addi $t57, $t49, $t49
69  addi $t58, $t50, $t50
70  addi $t59, $t51, $t51
71  addi $t60, $t52, $t52
72  addi $t61, $t53, $t53
73  addi $t62, $t54, $t54
74  addi $t63, $t55, $t55
75  addi $t64, $t56, $t56
76  addi $t65, $t57, $t57
77  addi $t66, $t58, $t58
78  addi $t67, $t59, $t59
79  addi $t68, $t60, $t60
80  addi $t69, $t61, $t61
81  addi $t70, $t62, $t62
82  addi $t71, $t63, $t63
83  addi $t72, $t64, $t64
84  addi $t73, $t65, $t65
85  addi $t74, $t66, $t66
86  addi $t75, $t67, $t67
87  addi $t76, $t68, $t68
88  addi $t77, $t69, $t69
89  addi $t78, $t70, $t70
90  addi $t79, $t71, $t71
91  addi $t80, $t72, $t72
92  addi $t81, $t73, $t73
93  addi $t82, $t74, $t74
94  addi $t83, $t75, $t75
95  addi $t84, $t76, $t76
96  addi $t85, $t77, $t77
97  addi $t86, $t78, $t78
98  addi $t87, $t79, $t79
99  addi $t88, $t80, $t80
100  addi $t89, $t81, $t81
101  addi $t90, $t82, $t82
102  addi $t91, $t83, $t83
103  addi $t92, $t84, $t84
104  addi $t93, $t85, $t85
105  addi $t94, $t86, $t86
106  addi $t95, $t87, $t87
107  addi $t96, $t88, $t88
108  addi $t97, $t89, $t89
109  addi $t98, $t90, $t90
110  addi $t99, $t91, $t91
111  addi $t100, $t92, $t92
112  addi $t101, $t93, $t93
113  addi $t102, $t94, $t94
114  addi $t103, $t95, $t95
115  addi $t104, $t96, $t96
116  addi $t105, $t97, $t97
117  addi $t106, $t98, $t98
118  addi $t107, $t99, $t99
119  addi $t108, $t100, $t100
120  addi $t109, $t101, $t101
121  addi $t110, $t102, $t102
122  addi $t111, $t103, $t103
123  addi $t112, $t104, $t104
124  addi $t113, $t105, $t105
125  addi $t114, $t106, $t106
126  addi $t115, $t107, $t107
127  addi $t116, $t108, $t108
128  addi $t117, $t109, $t109
129  addi $t118, $t110, $t110
130  addi $t119, $t111, $t111
131  addi $t120, $t112, $t112
132  addi $t121, $t113, $t113
133  addi $t122, $t114, $t114
134  addi $t123, $t115, $t115
135  addi $t124, $t116, $t116
136  addi $t125, $t117, $t117
137  addi $t126, $t118, $t118
138  addi $t127, $t119, $t119
139  addi $t128, $t120, $t120
140  addi $t129, $t121, $t121
141  addi $t130, $t122, $t122
142  addi $t131, $t123, $t123
143  addi $t132, $t124, $t124
144  addi $t133, $t125, $t125
145  addi $t134, $t126, $t126
146  addi $t135, $t127, $t127
147  addi $t136, $t128, $t128
148  addi $t137, $t129, $t129
149  addi $t138, $t130, $t130
150  addi $t139, $t131, $t131
151  addi $t140, $t132, $t132
152  addi $t141, $t133, $t133
153  addi $t142, $t134, $t134
154  addi $t143, $t135, $t135
155  addi $t144, $t136, $t136
156  addi $t145, $t137, $t137
157  addi $t146, $t138, $t138
158  addi $t147, $t139, $t139
159  addi $t148, $t140, $t140
160  addi $t149, $t141, $t141
161  addi $t150, $t142, $t142
162  addi $t151, $t143, $t143
163  addi $t152, $t144, $t144
164  addi $t153, $t145, $t145
165  addi $t154, $t146, $t146
166  addi $t155, $t147, $t147
167  addi $t156, $t148, $t148
168  addi $t157, $t149, $t149
169  addi $t158, $t150, $t150
170  addi $t159, $t151, $t151
171  addi $t160, $t152, $t152
172  addi $t161, $t153, $t153
173  addi $t162, $t154, $t154
174  addi $t163, $t155, $t155
175  addi $t164, $t156, $t156
176  addi $t165, $t157, $t157
177  addi $t166, $t158, $t158
178  addi $t167, $t159, $t159
179  addi $t168, $t160, $t160
180  addi $t169, $t161, $t161
181  addi $t170, $t162, $t162
182  addi $t171, $t163, $t163
183  addi $t172, $t164, $t164
184  addi $t173, $t165, $t165
185  addi $t174, $t166, $t166
186  addi $t175, $t167, $t167
187  addi $t176, $t168, $t168
188  addi $t177, $t169, $t169
189  addi $t178, $t170, $t170
190  addi $t179, $t171, $t171
191  addi $t180, $t172, $t172
192  addi $t181, $t173, $t173
193  addi $t182, $t174, $t174
194  addi $t183, $t175, $t175
195  addi $t184, $t176, $t176
196  addi $t185, $t177, $t177
197  addi $t186, $t178, $t178
198  addi $t187, $t179, $t179
199  addi $t188, $t180, $t180
200  addi $t189, $t181, $t181
201  addi $t190, $t182, $t182
202  addi $t191, $t183, $t183
203  addi $t192, $t184, $t184
204  addi $t193, $t185, $t185
205  addi $t194, $t186, $t186
206  addi $t195, $t187, $t187
207  addi $t196, $t188, $t188
208  addi $t197, $t189, $t189
209  addi $t198, $t190, $t190
210  addi $t199, $t191, $t191
211  addi $t200, $t192, $t192
212  addi $t201, $t193, $t193
213  addi $t202, $t194, $t194
214  addi $t203, $t195, $t195
215  addi $t204, $t196, $t196
216  addi $t205, $t197, $t197
217  addi $t206, $t198, $t198
218  addi $t207, $t199, $t199
219  addi $t208, $t200, $t200
220  addi $t209, $t201, $t201
221  addi $t210, $t202, $t202
222  addi $t211, $t203, $t203
223  addi $t212, $t204, $t204
224  addi $t213, $t205, $t205
225  addi $t214, $t206, $t206
226  addi $t215, $t207, $t207
227  addi $t216, $t208, $t208
228  addi $t217, $t209, $t209
229  addi $t218, $t210, $t210
230  addi $t219, $t211, $t211
231  addi $t220, $t212, $t212
232  addi $t221, $t213, $t213
233  addi $t222, $t214, $t214
234  addi $t223, $t215, $t215
235  addi $t224, $t216, $t216
236  addi $t225, $t217, $t217
237  addi $t226, $t218, $t218
238  addi $t227, $t219, $t219
239  addi $t228, $t220, $t220
240  addi $t229, $t221, $t221
241  addi $t230, $t222, $t222
242  addi $t231, $t223, $t223
243  addi $t232, $t224, $t224
244  addi $t233, $t225, $t225
245  addi $t234, $t226, $t226
246  addi $t235, $t227, $t227
247  addi $t236, $t228, $t228
248  addi $t237, $t229, $t229
249  addi $t238, $t230, $t230
250  addi $t239, $t231, $t231
251  addi $t240, $t232, $t232
252  addi $t241, $t233, $t233
253  addi $t242, $t234, $t234
254  addi $t243, $t235, $t235
255  addi $t244, $t236, $t236
256  addi $t245, $t237, $t237
257  addi $t246, $t238, $t238
258  addi $t247, $t239, $t239
259  addi $t248, $t240, $t240
260  addi $t249, $t241, $t241
261  addi $t250, $t242, $t242
262  addi $t251, $t243, $t243
263  addi $t252, $t244, $t244
264  addi $t253, $t245, $t245
265  addi $t254, $t246, $t246
266  addi $t255, $t247, $t247
267  addi $t256, $t248, $t248
268  addi $t257, $t249, $t249
269  addi $t258, $t250, $t250
270  addi $t259, $t251, $t251
271  addi $t260, $t252, $t252
272  addi $t261, $t253, $t253
273  addi $t262, $t254, $t254
274  addi $t263, $t255, $t255
275  addi $t264, $t256, $t256
276  addi $t265, $t257, $t257
277  addi $t266, $t258, $t258
278  addi $t267, $t259, $t259
279  addi $t268, $t260, $t260
280  addi $t269, $t261, $t261
281  addi $t270, $t262, $t262
282  addi $t271, $t263, $t263
283  addi $t272, $t264, $t264
284  addi $t273, $t265, $t265
285  addi $t274, $t266, $t266
286  addi $t275, $t267, $t267
287  addi $t276, $t268, $t268
288  addi $t277, $t269, $t269
289  addi $t278, $t270, $t270
290  addi $t279, $t271, $t271
291  addi $t280, $t272, $t272
292  addi $t281, $t273, $t273
293  addi $t282, $t274, $t274
294  addi $t283, $t275, $t275
295  addi $t284, $t276, $t276
296  addi $t285, $t277, $t277
297  addi $t286, $t278, $t278
298  addi $t287, $t279, $t279
299  addi $t288, $t280, $t280
300  addi $t289, $t281, $t281
301  addi $t290, $t282, $t282
302  addi $t291, $t283, $t283
303  addi $t292, $t284, $t284
304  addi $t293, $t285, $t285
305  addi $t294, $t286, $t286
306  addi $t295, $t287, $t287
307  addi $t296, $t288, $t288
308  addi $t297, $t289, $t289
309  addi $t298, $t290, $t290
310  addi $t299, $t291, $t291
311  addi $t300, $t292, $t292
312  addi $t301, $t293, $t293
313  addi $t302, $t294, $t294
314  addi $t303, $t295, $t295
315  addi $t304, $t296, $t296
316  addi $t305, $t297, $t297
317  addi $t306, $t298, $t298
318  addi $t307, $t299, $t299
319  addi $t308, $t300, $t300
320  addi $t309, $t301, $t301
321  addi $t310, $t302, $t302
322  addi $t311, $t303, $t303
323  addi $t312, $t304, $t304
324  addi $t313, $t305, $t305
325  addi $t314, $t306, $t306
326  addi $t315, $t307, $t307
327  addi $t316, $t308, $t308
328  addi $t317, $t309, $t309
329  addi $t318, $t310, $t310
330  addi $t319, $t311, $t311
331  addi $t320, $t312, $t312
332  addi $t321, $t313, $t313
333  addi $t322, $t314, $t314
334  addi $t323, $t315, $t315
335  addi $t324, $t316, $t316
336  addi $t325, $t317, $t317
337  addi $t326, $t318, $t318
338  addi $t327, $t319, $t319
339  addi $t328, $t320, $t320
340  addi $t329, $t321, $t321
341  addi $t330, $t322, $t322
342  addi $t331, $t323, $t323
343  addi $t332, $t324, $t324
344  addi $t333, $t325, $t325
345  addi $t334, $t326, $t326
346  addi $t335, $t327, $t327
347  addi $t336, $t328, $t328
348  addi $t337, $t329, $t329
349  addi $t338, $t330, $t330
350  addi $t339, $t331, $t331
351  addi $t340, $t332, $t332
352  addi $t341, $t333, $t333
353  addi $t342, $t334, $t334
354  addi $t343, $t335, $t335
355  addi $t344, $t336, $t336
356  addi $t345, $t337, $t337
357  addi $t346, $t338, $t338
358  addi $t347, $t339, $t339
359  addi $t348, $t340, $t340
360  addi $t349, $t341, $t341
361  addi $t350, $t342, $t342
362  addi $t351, $t343, $t343
363  addi $t352, $t344, $t344
364  addi $t353, $t345, $t345
365  addi $t354, $t346, $t346
366  addi $t355, $t347, $t347
367  addi $t356, $t348, $t348
368  addi $t357, $t349, $t349
369  addi $t358, $t350, $t350
370  addi $t359, $t351, $t351
371  addi $t360, $t352, $t352
372  addi $t361, $t353, $t353
373  addi $t362, $t354, $t354
374  addi $t363, $t355, $t355
375  addi $t364, $t356, $t356
376  addi $t365, $t357, $t357
377  addi $t366, $t358, $t358
378  addi $t367, $t359, $t359
379  addi $t368, $t360, $t360
380  addi $t369, $t361, $t361
381  addi $t370, $t362, $t362
382  addi $t371, $t363, $t363
383  addi $t372, $t364, $t364
384  addi $t373, $t365, $t365
385  addi $t374, $t366, $t366
386  addi $t375, $t367, $t367
387  addi $t376, $t368, $t368
388  addi $t377, $t369, $t369
389  addi $t378, $t370, $t370
390  addi $t379, $t371, $t371
391  addi $t380, $t372, $t372
392  addi $t381, $t373, $t373
393  addi $t382, $t374, $t374
394  addi $t383, $t375, $t375
395  addi $t384, $t376, $t376
396  addi $t385, $t377, $t377
397  addi $t386, $t378, $t378
398  addi $t387, $t379, $t379
399  addi $t388, $t380, $t380
400  addi $t389, $t381, $t381
401  addi $t390, $t382, $t382
402  addi $t391, $t383, $t383
403  addi $t392, $t384, $t384
404  addi $t393, $t385, $t385
405  addi $t394, $t386, $t386
406  addi $t395, $t387, $t387
407  addi $t396, $t388, $t388
408  addi $t397, $t389, $t389
409  addi $t398, $t390, $t390
410  addi $t399, $t391, $t391
411  addi $t400, $t392, $t392
412  addi $t401, $t393, $t393
413  addi $t402, $t394, $t394
414  addi $t403, $t395, $t395
415  addi $t404, $t396, $t396
416  addi $t405, $t397, $t397
417  addi $t406, $t398, $t398
418  addi $t407, $t399, $t399
419  addi $t408, $t400, $t400
420  addi $t409, $t401, $t401
421  addi $t410, $t402, $t402
422  addi $t411, $t403, $t403
423  addi $t412, $t404, $t404
424  addi $t413, $t405, $t405
425  addi $t414, $t406, $t406
426  addi $t415, $t407, $t407
427  addi $t416, $t408, $t408
428  addi $t417, $t409, $t409
429  addi $t418, $t410, $t410
430  addi $t419, $t411, $t411
431  addi $t420, $t412, $t412
432  addi $t421, $t413, $t413
433  addi $t422, $t414, $t414
434  addi $t423, $t415, $t415
435  addi $t424, $t416, $t416
436  addi $t425, $t417, $t417
437  addi $t426, $t418, $t418
438  addi $t427, $t419, $t419
439  addi $t428, $t420, $t420
440  addi $t429, $t421, $t421
441  addi $t430, $t422, $t422
442  addi $t431, $t423, $t423
443  addi $t432, $t424, $t424
444  addi $t433, $t425, $t425
445  addi $t434, $t426, $t426
446  addi $t435, $t427, $t427
447  addi $t436, $t428, $t428
448  addi $t437, $t429, $t429
449  addi $t438, $t430, $t430
450  addi $t439, $t431, $t431
451  addi $t440, $t432, $t432
452  addi $t441, $t433, $t433
453  addi $t442, $t434, $t434
454  addi $t443, $t435, $t435
455  addi $t444, $t436, $t436
456  addi $t445, $t437, $t437
457  addi $t446, $t438, $t438
458  addi $t447, $t439, $t439
459  addi $t448, $t440, $t440
460  addi $t449, $t441, $t441
461  addi $t450, $t442, $t442
462  addi $t451, $t443, $t443
463  addi $t452, $t444, $t444
464  addi $t453, $t445, $t445
465  addi $t454, $t446, $t446
466  addi $t455, $t447, $t447
467  addi $t456, $t448, $t448
468  addi $t457, $t449, $t449
469  addi $t458, $t450, $t450
470  addi $t459, $t451, $t451
471  addi $t460, $t452, $t452
472  addi $t461, $t453, $t453
473  addi $t462, $t454, $t454
474  addi $t463, $t455, $t455
475  addi $t464, $t456, $t456
476  addi $t465, $t457, $t457
477  addi $t466, $t458, $t458
478  addi $t467, $t459, $t459
479  addi $t468, $t460, $t460
480  addi $t469, $t461, $t461
481  addi $t470, $t462, $t462
482  addi $t471, $t463, $t463
483  addi $t472, $t464, $t464
484  addi $t473, $t465, $t465
485  addi $t474, $t466, $t466
486  addi $t475, $t467, $t467
487  addi $t476, $t468, $t468
488  addi $t477, $t469, $t469
489  addi $t478, $t470, $t470
490  addi $t479, $t471, $t471
491  addi $t480, $t472, $t472
492  addi $t481, $t473, $t473
493  addi $t482, $t474, $t474
494  addi $t483, $t475, $t475
495  addi $t484, $t476, $t476
496  addi $t485, $t477, $t477
497  addi $t486, $t478, $t478
498  addi $t487, $t479, $t479
499  addi $t488, $t480, $t480
500  addi $t489, $t481, $t481
501  addi $t490, $t482, $t482
502  addi $t491, $t483, $t483
503  addi $t492, $t484, $t484
504  addi $t493, $t485, $t485
505  addi $t494, $t486, $t486
506  addi $t495, $t487, $t487
507  addi $t496, $t488, $t488
508  addi $t497, $t489, $t489
509  addi $t498, $t490, $t490
510  addi $t499, $t491, $t491
511  addi $t500, $t492, $t492
512  addi $t501, $t493, $t493
513  addi $t502, $t494, $t494
514  addi $t503, $t495, $t495
515  addi $t504, $t496, $t496
516  addi $t505, $t497, $t497
517  addi $t506, $t498, $t498
518  addi $t507, $t499, $t499
519  addi $t508, $t500, $t500
520  addi $t509, $t501, $t501
521  addi $t510, $t502, $t502
522  addi $t511, $t503, $t503
523  addi $t512, $t504, $t504
524  addi $t513, $t505, $t505
525  addi $
```

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.



A screenshot of a debugger interface. On the left is the assembly code window, showing a series of machine code instructions. On the right is the registers window, showing various CPU register values. The assembly code includes labels like `start:`, `loop:`, and `end:`, along with various arithmetic and logical operations.

Loops & Jumps in Machine Language

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



A screenshot of a hex editor application. The left pane shows a list of memory pages, with the first page selected. The right pane displays assembly code in a text-based interface. The code consists of several lines of assembly instructions, likely demonstrating a loop or jump operation. The assembly language appears to be Intel 8086/32-bit syntax.

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

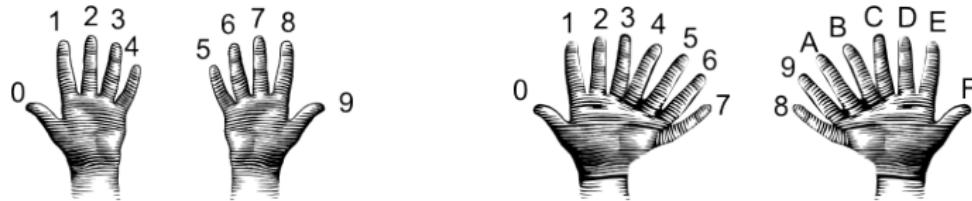


Today's Topics



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

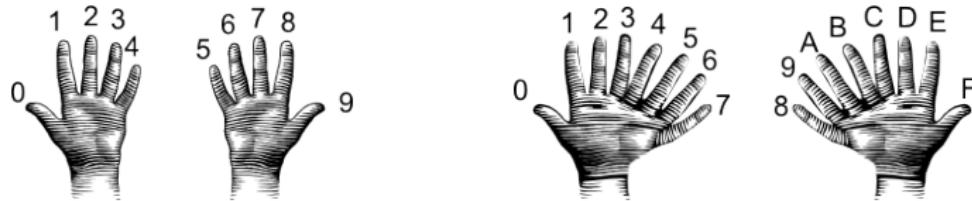
Hexadecimal to Decimal: Converting Between Bases



(from i-programmer.info)

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

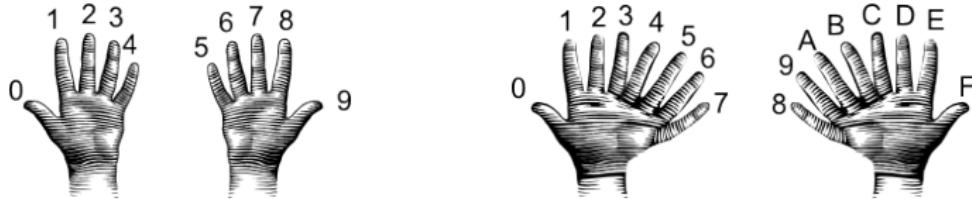
Hexadecimal to Decimal: Converting Between Bases



(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.

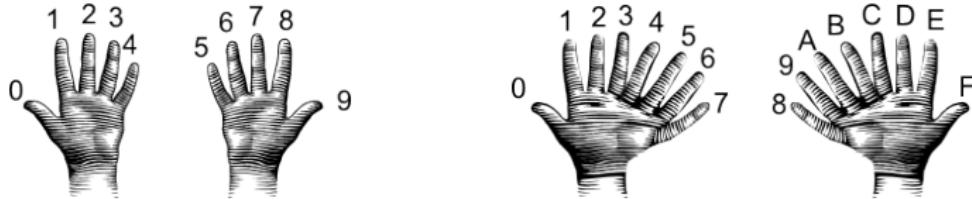
Hexadecimal to Decimal: Converting Between Bases



(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?

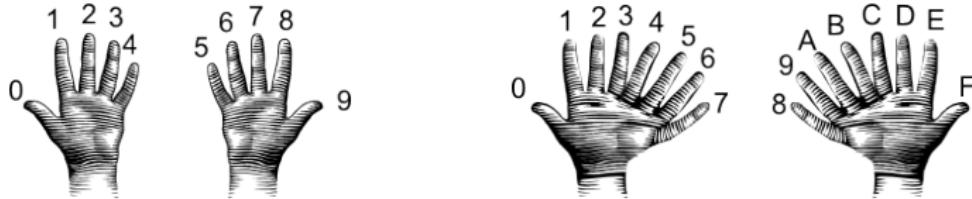
Hexadecimal to Decimal: Converting Between Bases



(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.

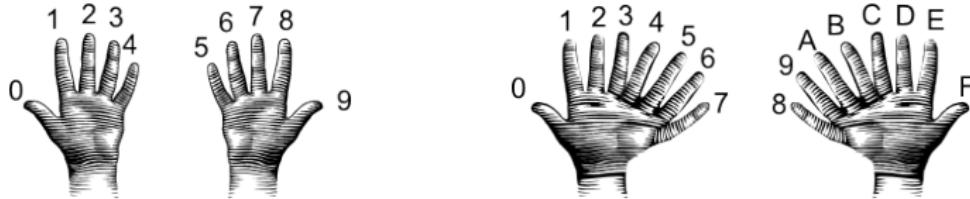
Hexadecimal to Decimal: Converting Between Bases



(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.

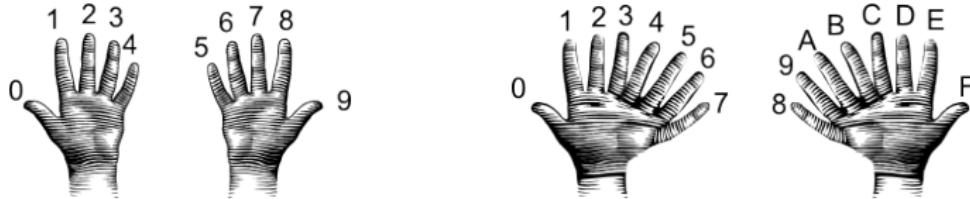
Hexadecimal to Decimal: Converting Between Bases



(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):

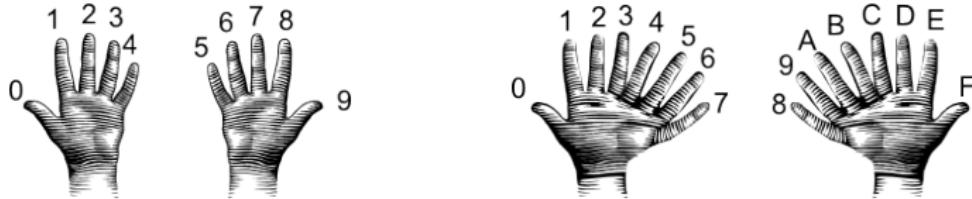
- Convert first digit to decimal and multiple by 16.
 - Convert second digit to decimal and add to total.
 - Example: what is 2A as a decimal number?

2 in decimal is 2. 2×16 is 32.

A in decimal digits is 10.

$32 + 10$ is 42.

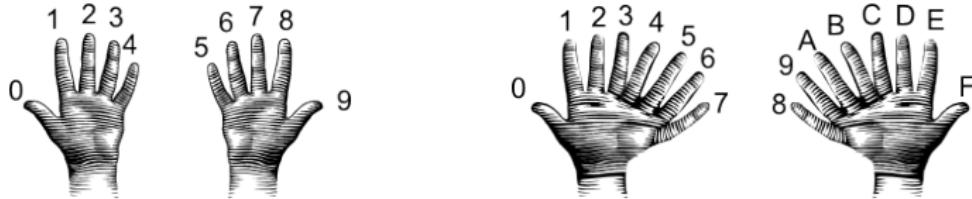
Hexadecimal to Decimal: Converting Between Bases



(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?

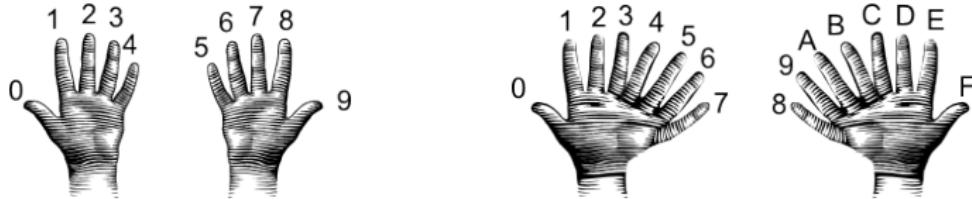
Hexadecimal to Decimal: Converting Between Bases



(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.

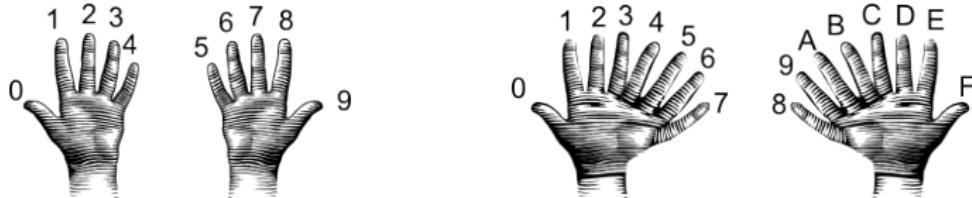
Hexadecimal to Decimal: Converting Between Bases



(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.

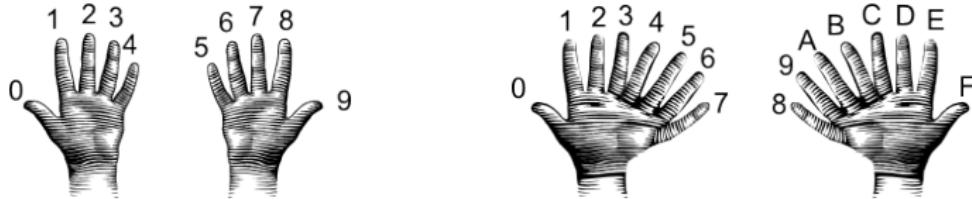
Hexadecimal to Decimal: Converting Between Bases



(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.

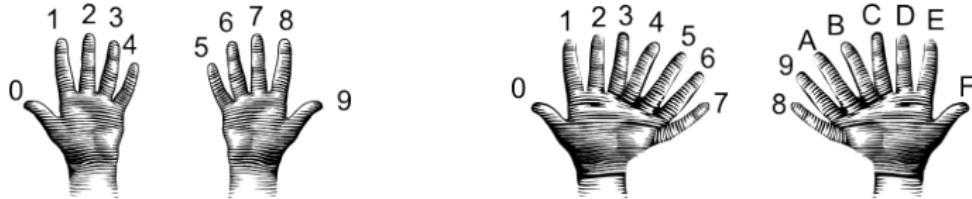
- Example: what is 99 as a decimal number?

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?

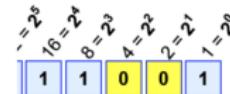
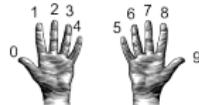
9 in decimal is 9. 9×16 is 144.

9 in decimal digits is 9

$144 + 9$ is 153.

Answer is 153.

Decimal to Binary: Converting Between Bases

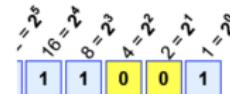
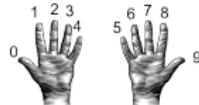


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

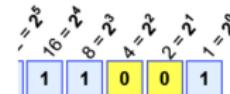
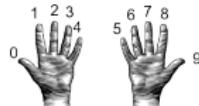


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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

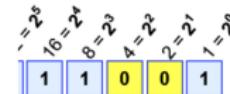
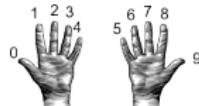


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 16 + 8 + 4 + 2 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

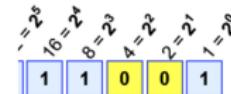


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = 16 + 8 + 4 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

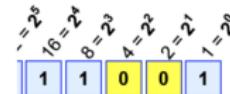
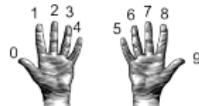


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

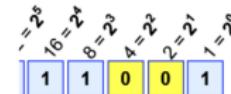
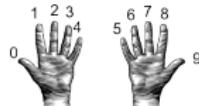


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

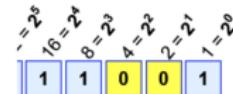
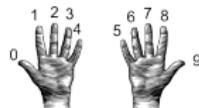


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

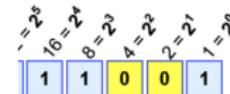
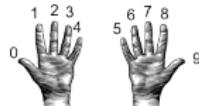


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- From decimal to binary:

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

Decimal to Binary: Converting Between Bases

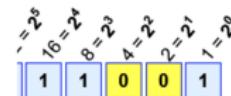
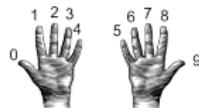


$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

- 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



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

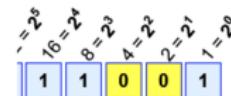
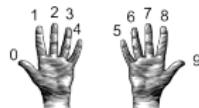
- From decimal to binary:

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

- Example: what is 130 in binary notation?

130/128 is 1 rem 2.

Decimal to Binary: Converting Between Bases



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

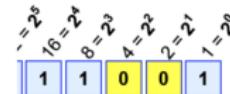
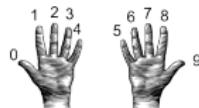
- From decimal to binary:

- 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



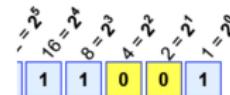
- From decimal to binary:

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

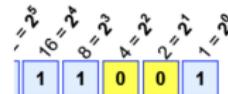
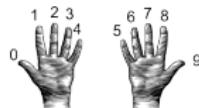
- From decimal to binary:

- 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



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

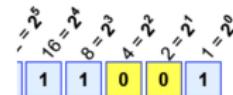
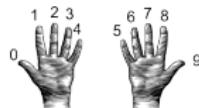
- From decimal to binary:

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

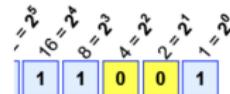
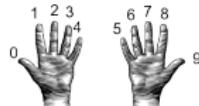
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

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

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

2/32 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

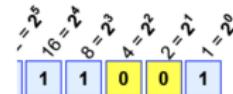
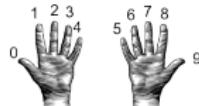
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

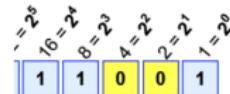
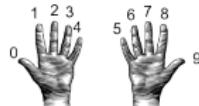
- Divide by $128 (= 2^7)$. Quotient is the first digit.
- Divide remainder by $64 (= 2^6)$. Quotient is the next digit.
- Divide remainder by $32 (= 2^5)$. Quotient is the next digit.
- Divide remainder by $16 (= 2^4)$. Quotient is the next digit.
- Divide remainder by $8 (= 2^3)$. Quotient is the next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

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

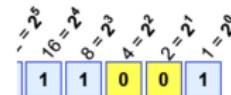
130/128 is 1 rem 2. First digit is 1: 1...

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

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

2/16 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

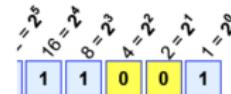
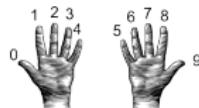
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

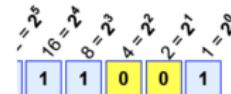
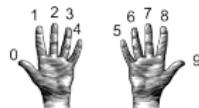
130/128 is 1 rem 2. First digit is 1: 1...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

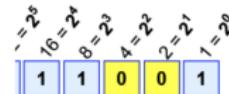
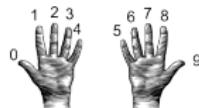
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

2/8 is 0 rem 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

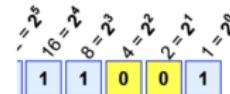
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

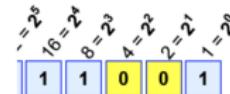
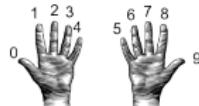
2/64 is 0 rem 2. Next digit is 0: 10...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

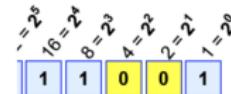
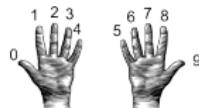
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

2/4 is 0 remainder 2.

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

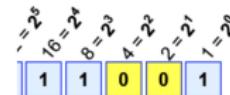
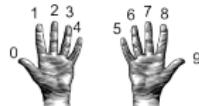
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

Decimal to Binary: Converting Between Bases



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

- From decimal to binary:

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

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

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

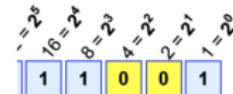
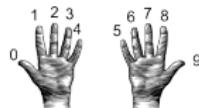
2/32 is 0 rem 2. Next digit is 0: 100...

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

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

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

Decimal to Binary: Converting Between Bases



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

- 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...

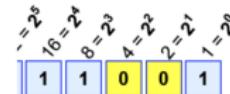
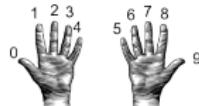
2/16 is 0 rem 2. Next digit is 0: 1000...

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



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

- From decimal to binary:

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

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

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

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

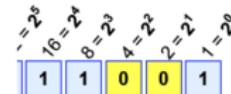
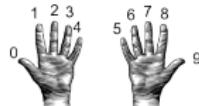
2/16 is 0 rem 2. Next digit is 0: 1000...

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

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

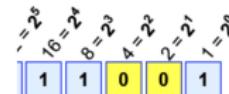
2/16 is 0 rem 2. Next digit is 0: 1000...

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...

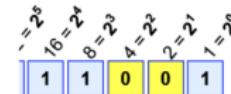
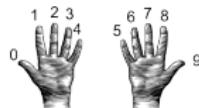
2/8 is 0 rem 2. Next digit is 0: 10000...

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...

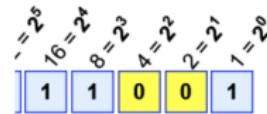
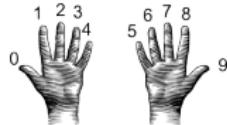
2/8 is 0 rem 2. Next digit is 0: 10000...

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

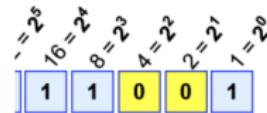
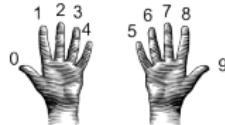
Decimal to Binary: Converting Between Bases



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

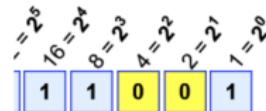


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

$99 / 128$ is 0 rem 99.

Decimal to Binary: Converting Between Bases

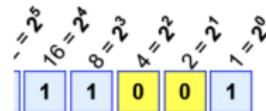


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: what is 99 in binary notation?

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

Decimal to Binary: Converting Between Bases



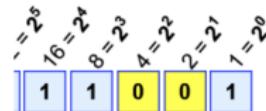
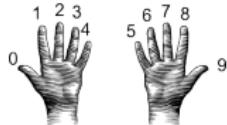
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



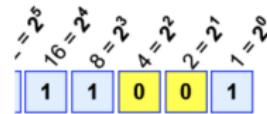
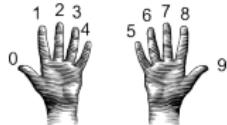
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



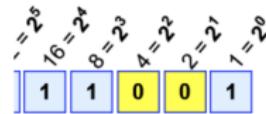
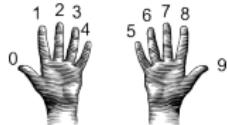
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

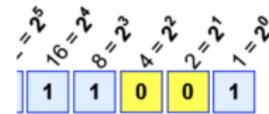
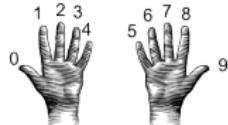
- Example: what is 99 in binary notation?

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

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

35/32 is 1 rem 3.

Decimal to Binary: Converting Between Bases



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

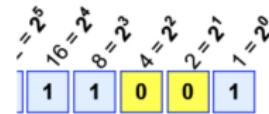
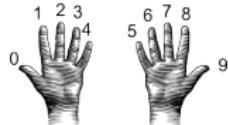
- Example: what is 99 in binary notation?

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

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

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

Decimal to Binary: Converting Between Bases



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

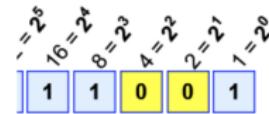
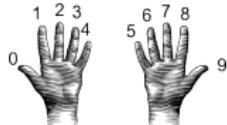
- Example: what is 99 in binary notation?

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

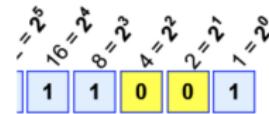
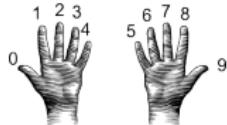
99/128 is 0 rem 99. First digit is 0: 0...

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

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

3/16 is 0 rem 3.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

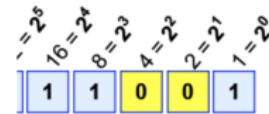
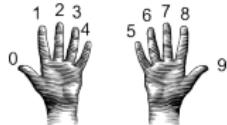
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

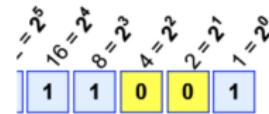
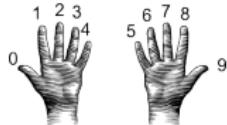
99/128 is 0 rem 99. First digit is 0: 0...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

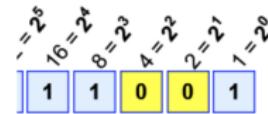
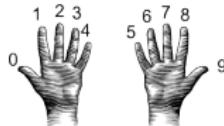
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

3/8 is 0 rem 3.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

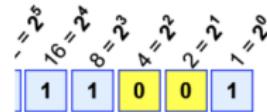
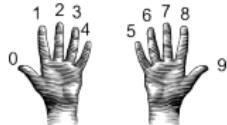
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

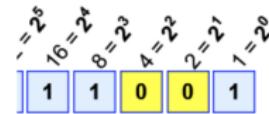
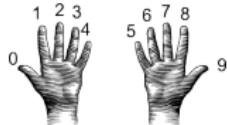
99/64 is 1 rem 35. Next digit is 1: 01...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

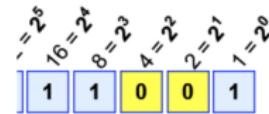
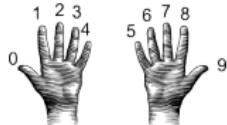
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

3/4 is 0 remainder 3.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

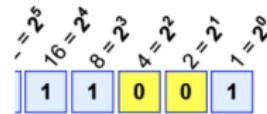
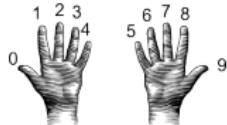
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

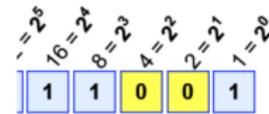
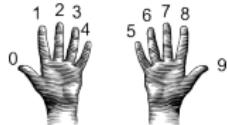
35/32 is 1 rem 3. Next digit is 1: 011...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

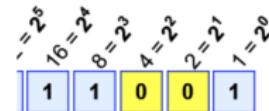
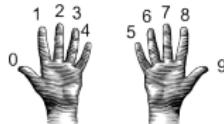
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

3/2 is 1 rem 1.

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

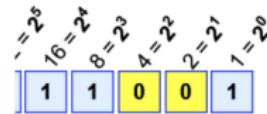
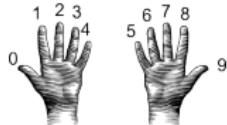
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

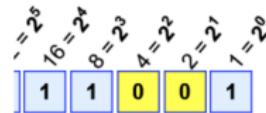
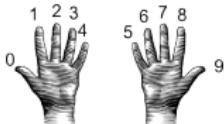
3/16 is 0 rem 3. Next digit is 0: 0110...

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

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

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

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

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

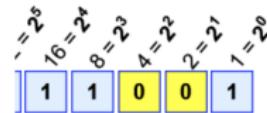
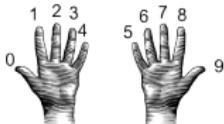
3/8 is 0 rem 3. Next digit is 0: 01100...

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

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

Adding the last remainder: 01100011

Decimal to Binary: Converting Between Bases



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

- Example: what is 99 in binary notation?

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

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

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

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

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

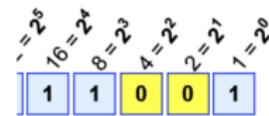
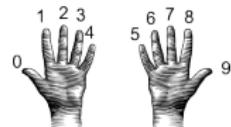
3/4 is 0 remainder 3. Next digit is 0: 011000...

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

Adding the last remainder: 01100011

Answer is 1100011.

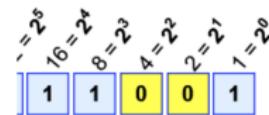
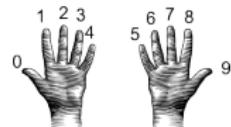
Binary to Decimal: Converting Between Bases



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

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

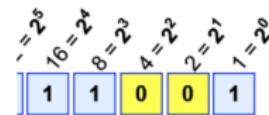
Binary to Decimal: Converting Between Bases



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

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

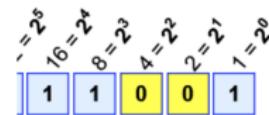
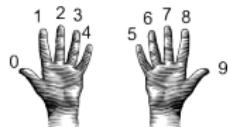
Binary to Decimal: Converting Between Bases



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

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

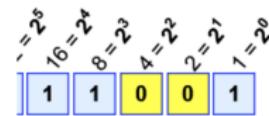
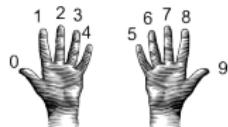
Binary to Decimal: Converting Between Bases



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

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

Binary to Decimal: Converting Between Bases

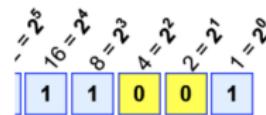
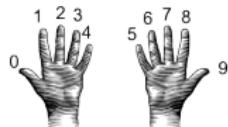


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

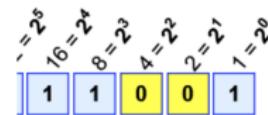
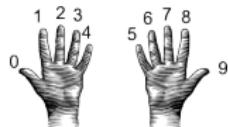


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

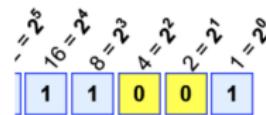
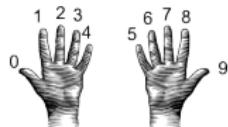


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

Binary to Decimal: Converting Between Bases

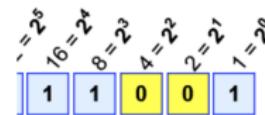
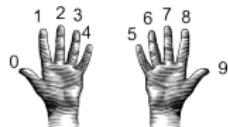


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases

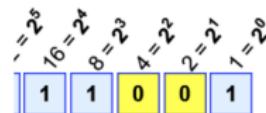


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

Binary to Decimal: Converting Between Bases



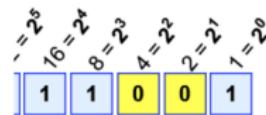
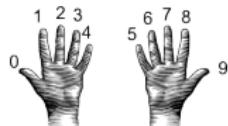
Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- From binary to decimal:

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

Sum starts with:

Binary to Decimal: Converting Between Bases



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

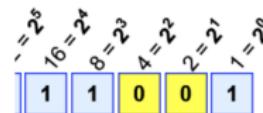
- From binary to decimal:

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

Sum starts with: 1

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

Binary to Decimal: Converting Between Bases



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

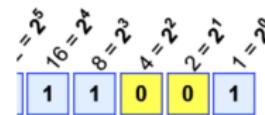
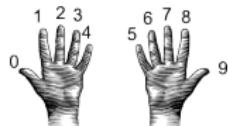
- From binary to decimal:

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

Sum starts with: 1

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

Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

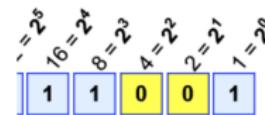
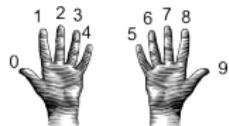
- Set sum = last digit.
- Multiply next digit by $2 = 2^1$. Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$.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

Sum starts with: 1

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

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

Binary to Decimal: Converting Between Bases



- From binary to decimal:

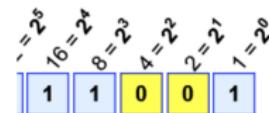
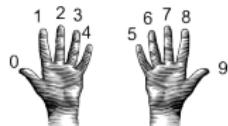
- Set sum = last digit.
- Multiply next digit by $2 = 2^1$. Add to sum.
- Multiply next digit by $4 = 2^2$. Add to sum.
- Multiply next digit by $8 = 2^3$. Add to sum.
- Multiply next digit by $16 = 2^4$. Add to sum.
- Multiply next digit by $32 = 2^5$. Add to sum.
- Multiply next digit by $64 = 2^6$.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

Sum starts with: 1

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

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

Binary to Decimal: Converting Between Bases



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

- From binary to decimal:

- Set sum = last digit.
- Multiply next digit by 2^1 . 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?

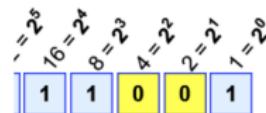
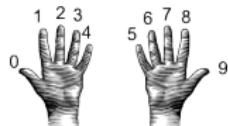
Sum starts with: 1

$0 \times 2 = 0$. 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



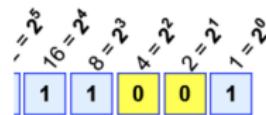
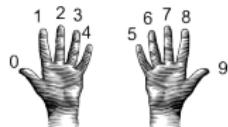
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



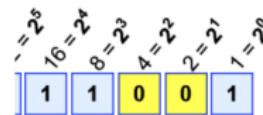
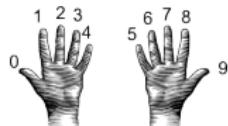
Example: $1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



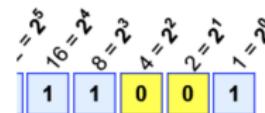
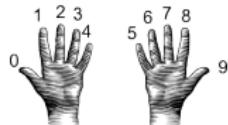
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



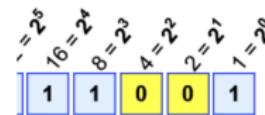
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



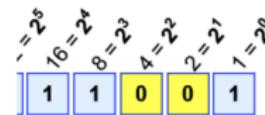
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases



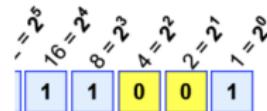
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- From binary to decimal:

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

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

Binary to Decimal: Converting Between Bases

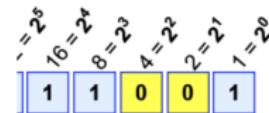


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- Example: What is 10100100 in decimal?

Sum starts with:

Binary to Decimal: Converting Between Bases



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

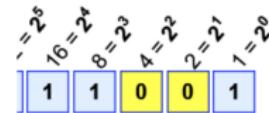
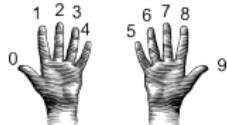
- Example: What is 10100100 in decimal?

Sum starts with:

0

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

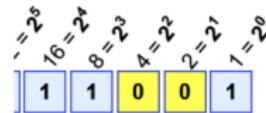
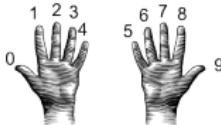
Sum starts with:

0

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

0

Binary to Decimal: Converting Between Bases



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

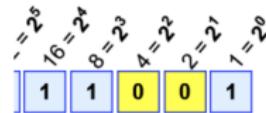
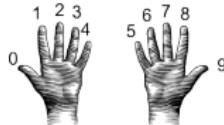
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

Binary to Decimal: Converting Between Bases



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

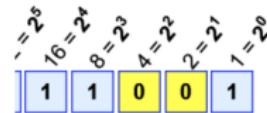
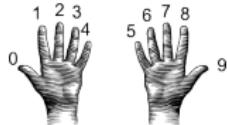
- Example: What is 10100100 in decimal?

Sum starts with: 0

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

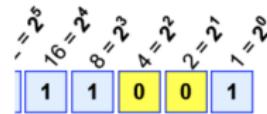
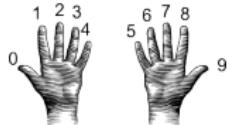
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

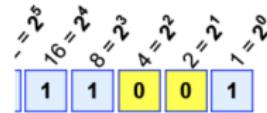
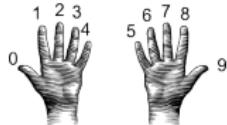
Sum starts with: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

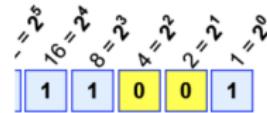
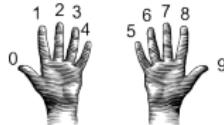
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

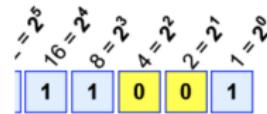
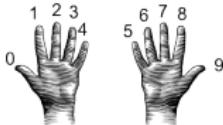
$0 \times 2 = 0$. Add 0 to sum: 0

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

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

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

Binary to Decimal: Converting Between Bases



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

- Example: What is 10100100 in decimal?

Sum starts with: 0

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

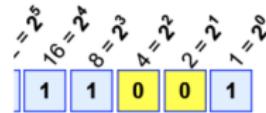
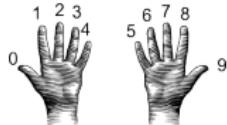
$1 \times 4 = 4$. Add 4 to sum: 4

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

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

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

Binary to Decimal: Converting Between Bases

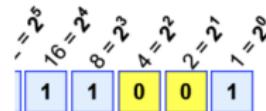
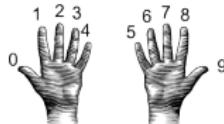


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

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

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

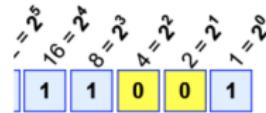
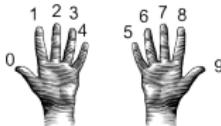
$0 \times 8 = 0$. Add 0 to sum: 4

$0 \times 16 = 0$. 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

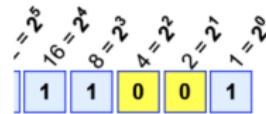
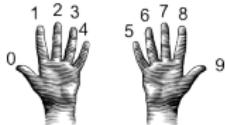
$0 \times 8 = 0$. Add 0 to sum: 4

$0 \times 16 = 0$. 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

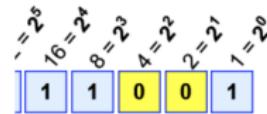
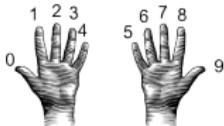
$0 \times 16 = 0$. Add 0 to sum: 4

$1 \times 32 = 32$. 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

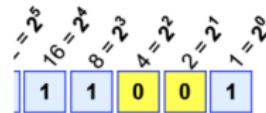


Example: $1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 16 + 8 + 4 + 0 + 1 = 25$

- Example: What is 10100100 in decimal?

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

Binary to Decimal: Converting Between Bases



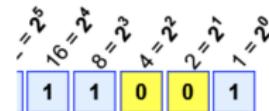
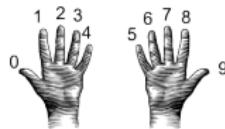
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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.

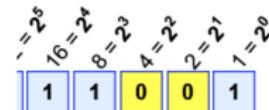
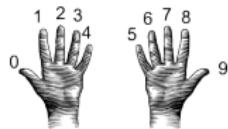
Design Challenge: Incrementers



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

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

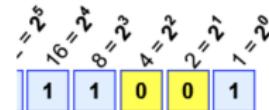
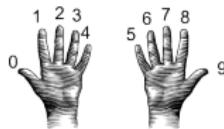
Design Challenge: Incrementers



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

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

Design Challenge: Incrementers

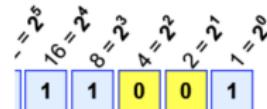
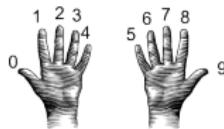


Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

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

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

Design Challenge: Incrementers



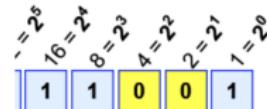
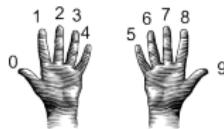
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



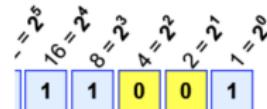
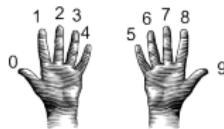
$$\text{Example: } 1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$$

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

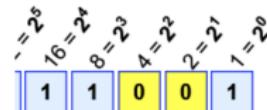
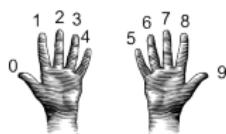
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
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



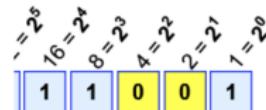
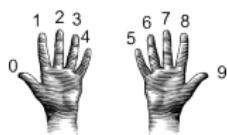
Example: $1 \times 16 + 1 \times 8 + 1 \times 1 = 16 + 8 + 1 = 25$

- 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



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

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

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

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

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

- Challenge: Write an algorithm for incrementing binary numbers.

Example: "1001" → "1010"

Recap



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

Recap



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

Today's Topics



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

Final Overview: Administration

- The exam will be administered through Gradescope.

Final Overview: Administration

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

Final Overview: Administration

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

Final Overview: Administration

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

Final Overview: Administration

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

Final Overview: Administration

- The exam will be administered through Gradescope.
- The exam will be available on Gradescope only on 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

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

18 December 2018

Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, among others) as serious violations of the Honor Code. Such acts are considered to be violations against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and punishes acts of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I acknowledge that all forms of academic dishonesty will be reported to the Honor Board and all grades in question.	
Name: _____	
SocSec# _____	
Email: _____	
Signature: _____	

Exam Options

Exam Times:

- Default Regular Time: Monday, 24 May, 9-11am.

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

18 December 2018

Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, among others) as serious violations of the Honor Code. Such acts are considered a violation against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and punishes acts of academic dishonesty according to the Hunter College Academic Integrity Procedure.

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

Exam Options

Exam Times:

- Default Regular Time: Monday, 24 May, 9-11am.
- Alternate Time: Friday, 21 May, 8am-10am.

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

19 December 2018

Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, among others) as serious violations of the Honor Code. Such acts are considered to be violations against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and pursuing cases of academic dishonesty according to the Hunter College Academic Integrity Procedure.

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

Exam Options

Exam Times:

- Default Regular Time: Monday, 24 May, 9-11am.
- Alternate Time: Friday, 21 May, 8am-10am.
- Accessibility Testing: If you have not done so already, email me no later than 7 May.

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

19 December 2018

Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, among others) as serious violations of the Honor Code. Such acts are considered a violation against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and punishes acts of academic dishonesty according to the Hunter College Academic Integrity Procedure.

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

Exam Options

Exam Times:

- Default Regular Time: Monday, 24 May, 9-11am.
- Alternate Time: Friday, 21 May, 8am-10am.
- Accessibility Testing: If you have not done so already, email me no later than 7 May.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on 24 May.**

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

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8.5" x 11" piece of paper that you can write on.
- When taking the exam, you may bring with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart phone, 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, among others) as serious violations of the Honor Code. Such acts will be referred to the Office of Student Conduct and will result in disciplinary action against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and pursues acts of academic dishonesty according to the Hunter College Academic Integrity Procedure.

I acknowledge that all forms of academic dishonesty will be reported to the Office of Student Conduct and will result in sanctions.
Name: _____
Student ID: _____
Email: _____
Signature: _____

Exam Options

Exam Times:

- Default Regular Time: Monday, 24 May, 9-11am.
- Alternate Time: Friday, 21 May, 8am-10am.
- Accessibility Testing: If you have not done so already, email me no later than 7 May.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on 24 May.**
- If you choose to take the early date, **you will not be given access to the exam on 24 May even if you miss the early exam.**

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

19 December 2018

Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, among others) as serious violations of the Honor Code. Such acts will be referred to the Office of Student Conduct and will result in sanctions against the violators of individual integrity. The College is committed to enforcing the CUNY Policy on Academic Integrity and to pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I acknowledge that all forms of academic dishonesty will be reported to the Office of Student Conduct and will result in sanctions.
Name: _____
Social ID: _____
Email: _____
Signature: _____

Exam Options

Exam Times:

- Default Regular Time: Monday, 24 May, 9-11am.
- Alternate Time: Friday, 21 May, 8am-10am.
- Accessibility Testing: If you have not done so already, email me no later than 7 May.
- Survey for your exam date choice will be available next lecture. **No survey answer implies you will take the exam on 24 May.**
- If you choose to take the early date, **you will not be given access to the exam on 24 May even if you miss the early exam.**

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

19 December 2018

Exam Rules

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

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating or communism, among others) as serious violations of its educational mission. Such acts are considered a violation against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and to pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

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

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- 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 Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 50-52)

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 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 Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 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)