Introduction to Programming for Scientists

Lecture 7

Prof. Steven Ludtke
N410, sludtke@bcm.edu

Homework Review

```
import os, sys
from PIL import Image
import ImageFilter
directory='G:/pictures'
files=os.listdir(directory)
for i in files:
    filepath=directory+'/'+i
    im=Image.open(filepath)
    filename,ext=os.path.splitext(filepath)
    im2=im.filter(ImageFilter.BLUR)
    im2.save(filename + ".png","PNG")
```

```
import glob
import Image
a=raw_input("x resize?")
b=raw_input("y resize?")
l=glob.glob('*.jpg')
for fname in l:
    Image.open(fname).resize((int(a),int(b))).save('%s.png'%fname[:-4])
```

pickle/shelve

- from pickle import dump,load,dumps,loads
- dump(obj,file) # stores 'obj' in 'file'
- obj=load(file) # restores 'obj' from file
- str=dumps(obj) # pickled representation of obj
- obj=loads(str) # restore representation of obj

- import shelve # dictionary-like object on disk
- dic=shelve.open(filename)
- dic=shelve.open(filename,writeback=True)
- dic.close()

numpy

- http://www.tramy.us/ # numpy book
- from numpy import *
- a=arange(60)
- b=a.reshape(10,6) # make 2-D matrix
- c=a.reshape(3,4,5) # make 3-D (tensor)
- b.shape # current dimensions
- b.size # total number of elements
- b.ndim # dimensionality
- b.dtype # type of value stored
- b.astype("")

Type	Bit-Width	Character
bool_	boolXX	'?'
byte	intXX	'b'
short		'h'
intc		'i'
int_		'l'
longlong		'q'
intp		'p'
ubyte	uintXX	'B'
ushort		'H'
uintc		'I'
uint		'L'
ulonglong		'Q'
uintp		'P'
single	floatXX	'f'
float_		'd'
longfloat		'g'
csingle	complexXX	'F'
complex_		'D'
clongfloat		'G'
object_		'O'
str_		'S#'
unicode_		'U#'
void		'V#'

numpy

- a=zeros((nx,ny,...))
- a=fromfunction(lambda i,j:i+j,(4,5))
- a=arange(0,20,.1) #
- a*10 # multiply each element !
- b=sin(a) # sin() of each element
- c=a[c>0] # condition, returns elements >0
- c.sort() # sort values in-place
- c.mean(),var(),std(),prod() # average, variance, standard dev, product
- inner(a,b), outer(a,b) # inner and outer matrix products
- dot(a,b), cross(a,b) # dot and cross products (similar to above)
- histogram(a,bins,range) # compute a histogram of 'a'
- from PIL import Image
- im=Image.fromarray(a,"L") # initialize a PIL image with a 2D array

scipy

- Clustering package (scipy.cluster)
- Constants (scipy.constants)
- Fourier transforms (scipy.fftpack)
- Integration and ODEs (scipy.integrate)
- Interpolation (scipy.interpolate)
- Input and output (scipy.io)
- Linear algebra (scipy.linalg)
- Maximum entropy models (scipy.maxentropy)
- Miscellaneous routines (scipy.misc)
- Multi-dimensional image processing (scipy.ndimage)
- Orthogonal distance regression (scipy.odr)
- Optimization and root finding (scipy.optimize)
- Signal processing (scipy.signal)
- Sparse matrices (scipy.sparse)
- Sparse linear algebra (scipy.sparse.linalg)
- Spatial algorithms and data structures (scipy.spatial)
- Special functions (scipy.special)
- Statistical functions (scipy.stats)
- Image Array Manipulation and Convolution (scipy.stsci)

matplotlib (pylab)

- Matlab-like plotting library
- http://matplotlib.sourceforge.net/users/pyplot_tutorial.html
- ipython --pylab # special mode for interaction with pylab

```
x=arange(0,4*pi,0.05)    # from numpy
y=sin(x)                 # easy to apply a function to a list of values
plot(x,y)                # plot x,y and open a display window
```

- python

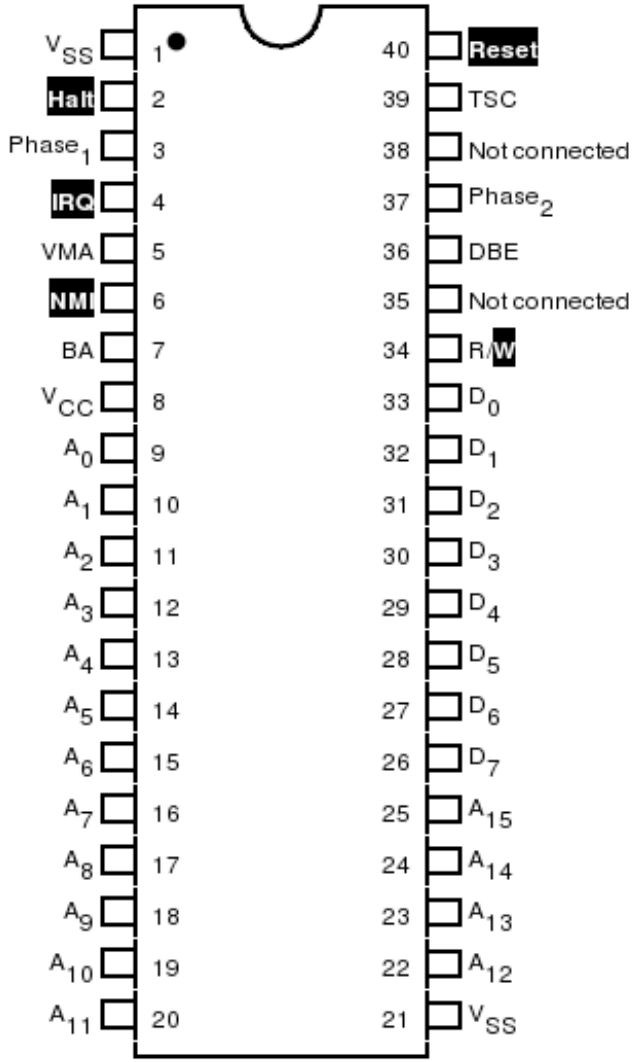
```
from pylab import *      # <-- only if you don't use ipython
x=arange(0,4*pi,0.05)
y=sin(x)                 # easy to apply a function to a list of values
plot(x,y)                # plot x,y and open a display window
show()                   # opens the plot window (blocks on some machines)
```

matplotlib (pylab)

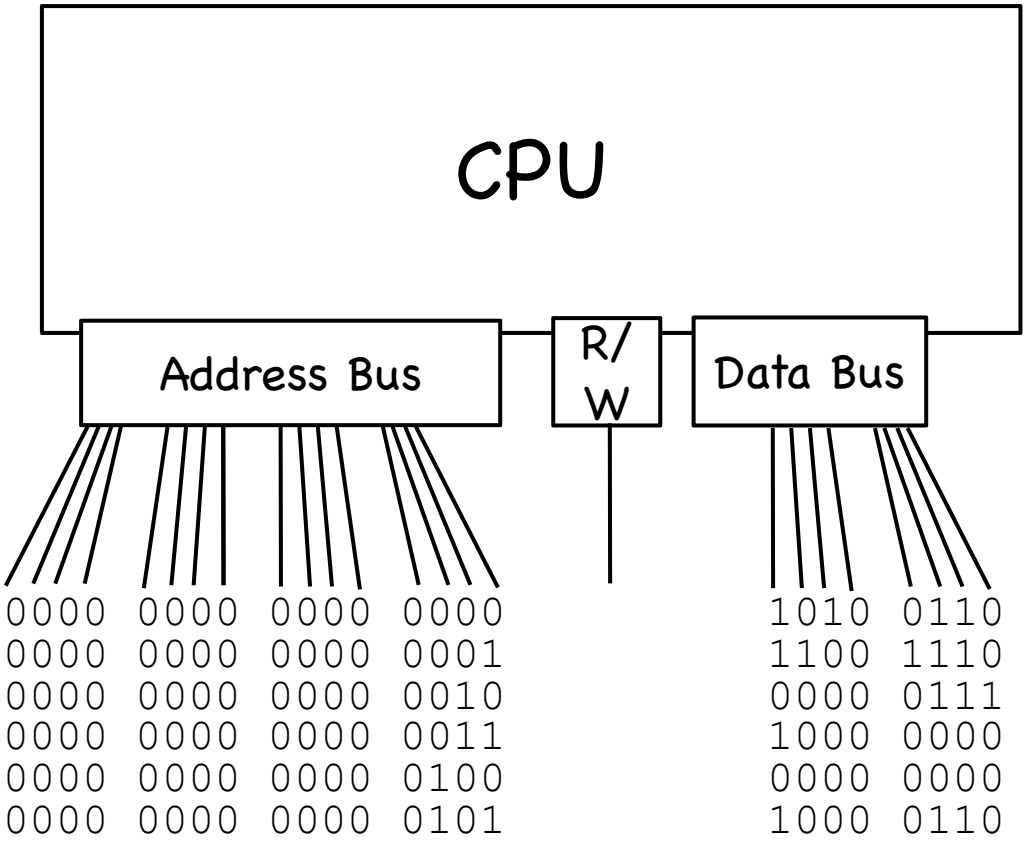
```
ipython --pylab          # special mode for interaction with pylab
x=arange(0,4*pi,0.05)    # from numpy
y=sin(x)                 # easy to apply a function to a list of values
y2=cos(x)
figure(1)                # start a new figure
subplot(211)             # make a 1x2 set of plots and move to 1st
plot(x,y)                # plot x,y and open a display window
ylabel("sin(x)")
subplot(212)             # start on the 2nd subplot
plot(x,y2)               # second plot
ylabel("cos(x)")
xlabel("x")
```

Motorola 6800 CPU

- 72 instructions (197 opcodes)
- 8 bit data bus (0-255)
- 16 bit address bus (64k max RAM)
- 6 registers:
  - 8 bit ACCA
  - 8 bit ACCB
  - 16 bit IX
  - 16 bit PC
  - 16 bit SP
  - 6 bit CC

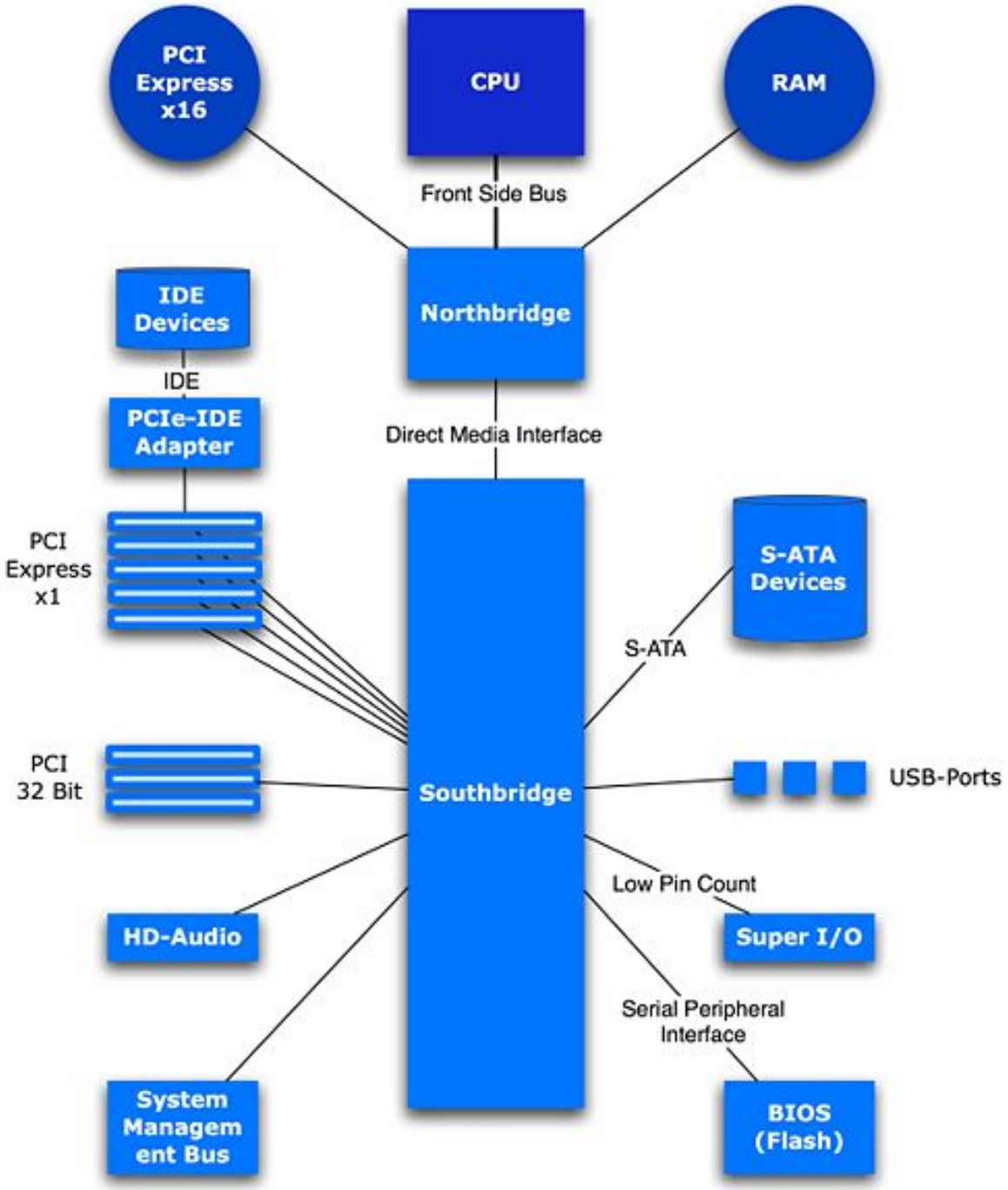


CPU Communications



Communications

- Are computers useful without i/o ?
  - KB/Mouse, video, sound, printer, digital camera, other computers, scanner, TV tuner, home control, specialized equipment (micromill, 3-D printer, telescope control, etc.), scientific instruments (!)
- Memory mapping
- Polling vs Interupts



http://commons.wikimedia.org/wiki/Image:Typical_intel_chipset.jpg

	Speed (Mb/s)	Range (m)	description	connector
PCIe (1x/16x)	250/4000	internal	16x mainly for video	
PCI (32/64)	133	internal	64 not widely used	
AGP (1x-8x)	266-2133*	internal	largely obsolete, for video	
network (10/100/1000)	1/12/120	100	1000=gigabit, 10G exists	
802.11 (a/b/g/n)	6/1/6/30	120//250	N is the new standard, most still use G	wireless
bluetooth	0.3	~10	short range device comm	wireless
IRDA	.01-1(?)	1	bluetooth instead	wireless
firewire/1394 (400/800)	50/100 (400?)	4.5/?	external disks, video	
USB (1/2/3*)	2/57/570	5/5	external disks, etc.	
Parallel	1	15	pre-USB for printers	
Serial	0.01		old-style modems, specialty devices	
PS/2	n/a		keyboard & mouse	

Disk Interfaces

	Speed (Mb/s)	connector
SATA	300/600	
IDE/ATA	3-133	
Fibre Channel	100-400	optical, long range (>50km)
SCSI	5-640	many connectors, external, short range
firewire/1394 (400/800)	50/100(400?)	
USB (1/2/3*)	2/57/570	
Actual Disk Perf.	80-150, 200-300	
Memory Bandwidth	~30,000	

Video Interfaces

	HDCP	Description	Connector
HDMI	X	High definition (digital) consumer video (and audio)	
DVI	X	Newer style computer video, digital, supports HDCP,	
VGA		Old style computer video, supports high resolution, but analog	
Component		Further improved quality, supports higher resolutions	
S-video		Improved quality, still 640x480	
Composite		Oldest style consumer video, poor quality, poor resolution (640x480)	

Homework 7

1. We're now going to take the program from homework 5 and make it work with matplotlib. You can use your original solution to homework 5 as a starting point. You have 4 tasks:

    1. Many of you didn't store the x/y data internally in the object you created. While this was ok for homework 5, please correct it for this homework assignment, as we are going to be extending this object in future homework assignments. That is, please 'fix' your object so the data is stored in the object and not taken as parameters to the various methods. Also make sure there is a separate method (other than just __init__) for reading data from disk.

    2. If neceesary, change the data storage of the x/y data from Python lists to use numpy arrays.

    3. Modify the histogram method so it makes use of matplotlib to display the results of the histogram graphically. Feel free to redo the histogram calculation using capabilities of numpy/scipy as well, though this is not required.

    4. Add an additional method to plot the x/y data as a 2-D line plot using matplotlib.