Lecture 01
Introduction to CSE 30321

Suggested reading:

None.

My contact information…

- Michael Niemier
  - (Mike is fine.)
- Contact information:
  - 380 Fitzpatrick Hall
  - mniemier@nd.edu
  - (574) 631-3858
- Office hours:
  - Mon: noon-1:30
  - Thurs: 2:30-4:00
- About me…

Fundamental lesson(s) for this lecture:

1. Why *Computer Architecture* is important for your major
   - (CS, CPEG, EE, MATH, etc…)
2. What should you be able to do when you finish class?

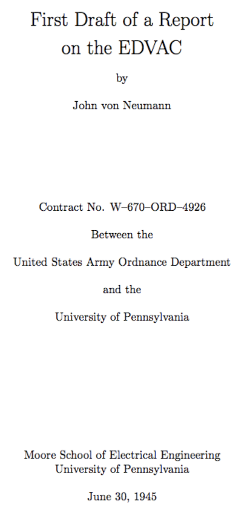
Why should I care?

- If you're interested in SW…
  - Understanding how processor HW operates, how it's organized, and the interface between HW and a HLL will make you a better programmer.
    - see "the matrix multiply" & "the 4- to 8-core" examples…
- If you're interested in HW…
  - Technology is rapidly changing how microprocessor architectures are designed and organized, and has introduced new metrics like "performance per Watt"
    - The material covered in this class is the baseline required to both understand and to help develop the "state of the art"
    - see "the technology drive to multi-core" material

How we process information hasn't changed much since 1930s and 1940s

A little history... programs

- Stored program model has been around for a long time...



First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W–670–ORD–4926

Between the

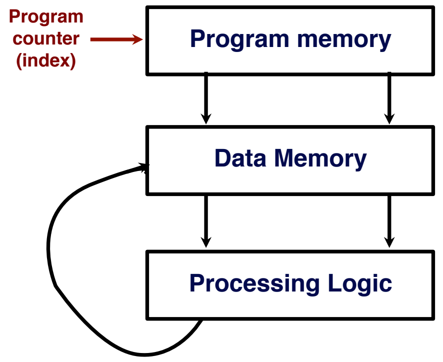
United States Army Ordnance Department

and the

University of Pennsylvania

Moore School of Electrical Engineering
University of Pennsylvania

June 30, 1945



Stored Program (continued)

A hypothetical translation:

```
for i=0; i<5; i++ {
     a = (a*b) + c;
}
```

MULT temp,a,b # temp ← a*b
MULT r1,r2,r3 # r1 ← r2*r3

ADD a,temp,c # a ← temp+c
ADD r2,r1,r4 # r2 ← r1+r4

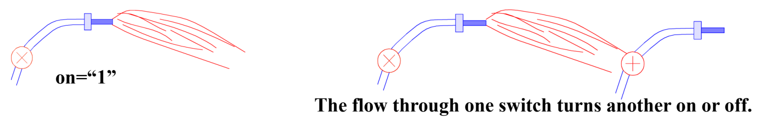
Can define codes for MULT and ADD
Assume MULT = 110011 & ADD = 001110

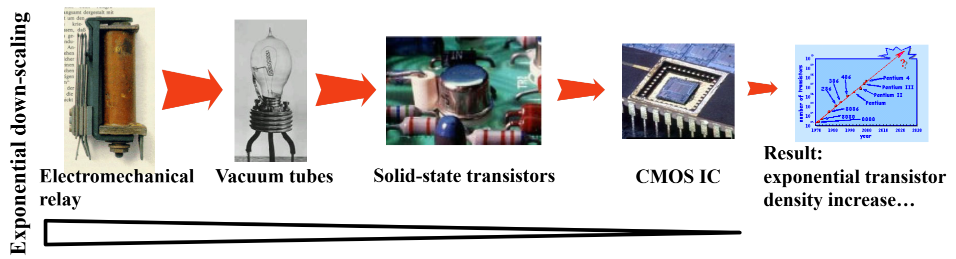
stored program becomes

PC	110011	000001	000010	000011
PC+1	001110	000010	000001	000100

A little history... Zuse's paradigm

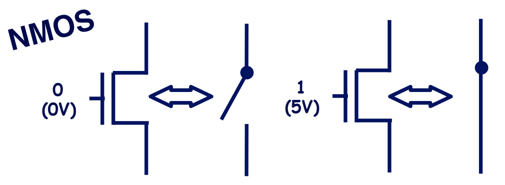
- Konrad Zuse (1938) Z3 machine
  - Use binary numbers to encode information
  - Represent binary digits as on/off state of a current switch



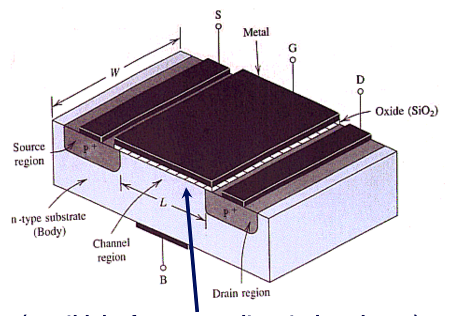


Transistors used to manipulate/store 1s & 0s

Switch-level representation



Cross-sectional view



(can think of as a capacitor storing charge)

If we (i) apply a suitable voltage to the gate & (ii) then apply a suitable voltage between source and drain, current will flow.

Moore's Law

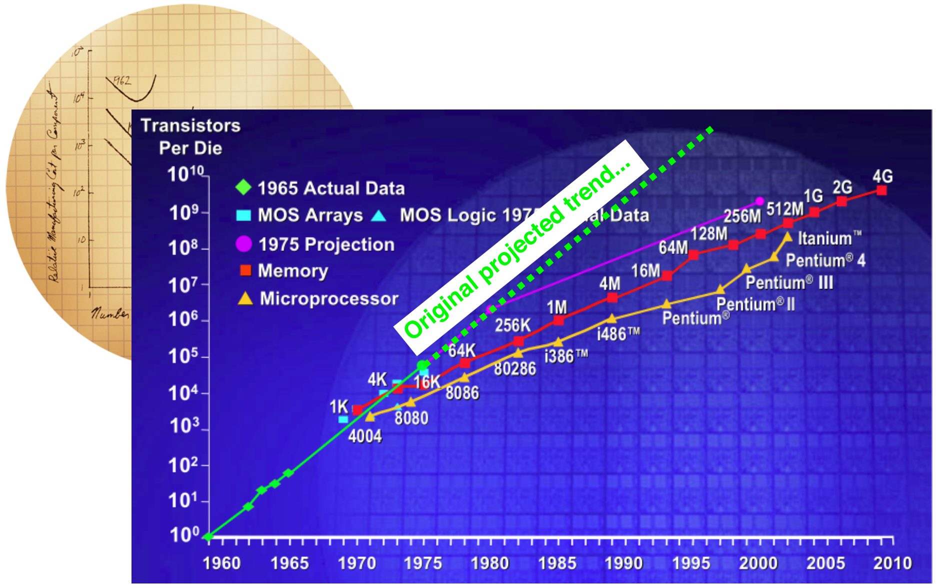
- "Cramming more components onto integrated circuits."

  - G.E. Moore, Electronics 1965

  - Observation: DRAM transistor density doubles annually
    - Became known as "Moore's Law"
    - Actually, a bit off:
      - Density doubles every 18 months (now more like 24)
      - (in 1965 they only had 4 data points!)
  - Corollaries:
    - Cost per transistor halves annually (18 months)
    - Power per transistor decreases with scaling
    - Speed increases with scaling
      - Of course, it depends on how small you try to make things
        - » (I.e. no exponential lasts forever)

Remember these!

Moore's Law



Feature sizes...

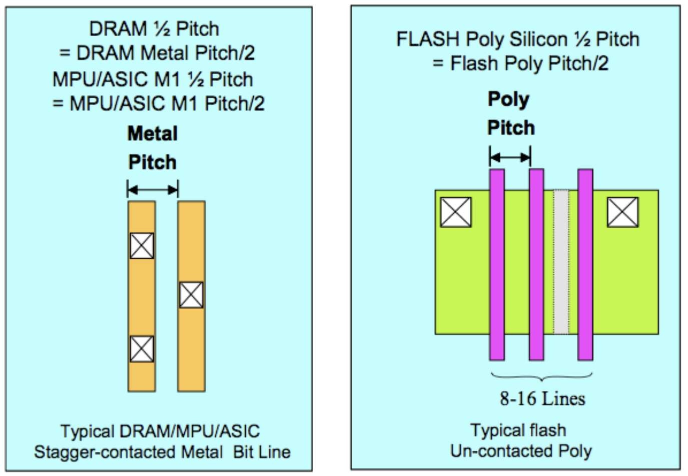


Figure 2 2005 Definition of Pitches

Moore's Law

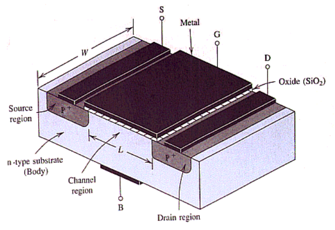
- Moore's Curve is a self-fulfilling prophecy
  - 2X every 2 years means ~3% per month
    - I.e. ((1 X 1.03) * 1.03)*1.03… 24 times = ~2
  - Can use 3% per month to judge performance features
  - If feature adds 9 months to schedule…it should add at least 30% to performance
    - ($1.03^9 = 1.30$ ⇨ 30%)

A bit on device performance...

- One way to think about switching time:
  - Charge is carried by electrons
  - Time for charge to cross channel = length/speed
    - $= L^2/(mV_{ds})$

Thus, to make a device faster, we want to either increase V_{ds} or decrease feature sizes (i.e. L)

- What about power (i.e. heat)?
  - Dynamic power is: $P_{dyn} = C_L V_{dd}^2 f_{0-1}$
    - $C_L = (e_{ox}WL)/d$
      - e_{ox} = dielectric, WL = parallel plate area, d = distance between gate and substrate



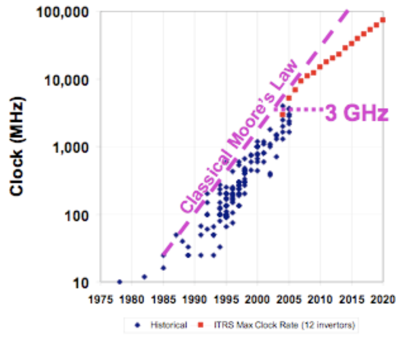
Summary of relationships

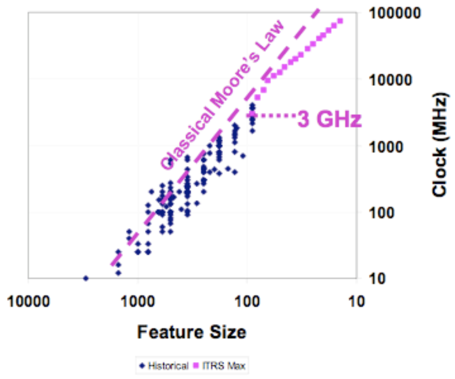
- (+) If V increases, speed (performance) increases
- (-) If V increases, power (heat) increases
- (+) If L decreases, speed (performance) increases
- (?) If L decreases, power (heat) does what?
  - P could improve because of lower C
  - P could increase because >> # of devices switch
  - P could increase because >> # of devices switch faster!

Need to carefully consider tradeoffs between speed and heat

So, what's happened?

•Speed increases with scaling...

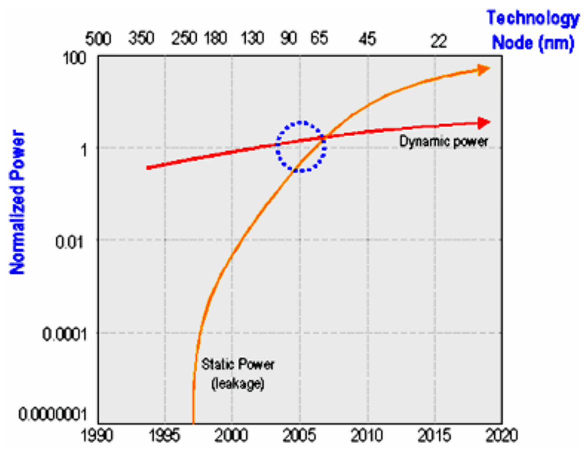




2005 projection was for 5.2 GHz - and we didn't make it in production. Further, we're still stuck at 3+ GHz in production.

So, what's happened?

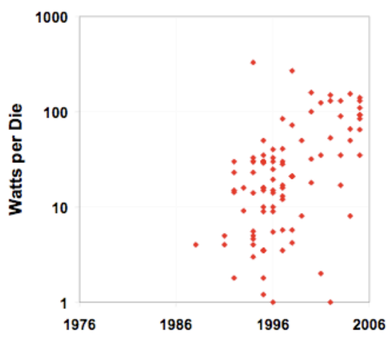
- Power decreases with scaling…



So, what's happened?

- Speed increases with scaling…
- Power decreases with scaling…

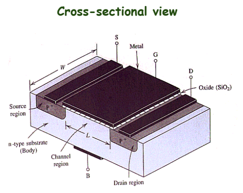
Why the clock flattening? POWER!!!!

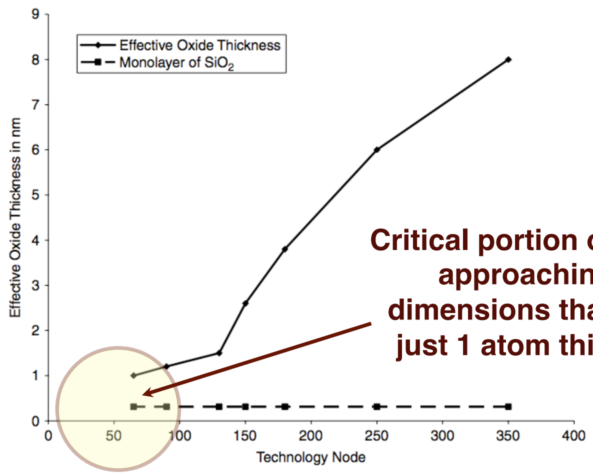


So, what's happened?

- What about scaling…

One quick example:





Critical portion of gate approaching dimensions that are just 1 atom thick!!!

Materials innovations were – and still are – needed

(Short term?) solution

High art meets high-tech.

Lincoln's latest project, titled "CUBE," is a 10' x 10' translucent structure outfitted with video cameras, uniquely combining sculpture, portraiture and architecture. With **Intel® Centrino® processor technology** inside, a notebook becomes many other things as well — portable studio, canvas, inspiration tool.

Top 5 Must-Haves

☐ **POWERFUL PROCESSOR**
A portrait of performance. "My generative portraits are demanding on the processors in my laptop, as they continuously manipulate video," says Lincoln. Thankfully, the **dual-core performance** of Intel Centrino processor technology can handle intensive tasks with flying colors.

☐ **DIZZYING TRANSFER SPEEDS**
Art (at 30 frames per second). Data transferring up to 20% faster[1] allows Lincoln to store footage from 24 video cameras with lightning speed

☐ **HIGH-SPEED WIRELESS**
Always Connected. With up to **twice the range** and **5x the speed** when connected to a Wireless N home network,[2] Lincoln can download music or shop for art books anywhere, anytime.

☐ **ENHANCED VIDEO**
High-def (redefined). Lincoln can view his generative portraits with "gallery-like" clarity, thanks to **stunning multimedia performance**, for a super-enhanced high-def video experience.

☐ **IMPROVED BATTERY LIFE**
The power of art. Lincoln's infinitely reconfiguring images are ultimately presented on a plasma screen powered by his computer — so wasting power is not an option. Thanks to **Intel's exclusive power-saving features**, he conserves energy by using it only when he needs it.

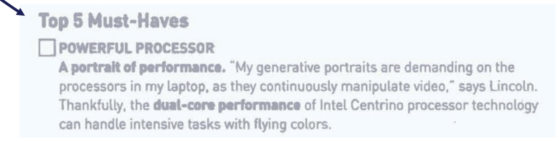
Deeper. Richer. Faster.
Log on to drivenbywhatsinside.com for access to exclusive multimedia content to keep you up-to-date on the latest tec trends — faster. To take advantage of this high-tech, multimedia material, make sure your computer has **Intel Centrino processor technology.**

intel Centrino

©2008 Intel Corporation. All rights reserved. Intel, the Intel l Centrino, and Centrino Inside are trademarks of Intel Corpora in the U.S. and other countries. [1]20% faster data transfer rate t previous-generation Intel Centrino processors. [2]Up to 2x gre range and up to 5x better performance and improved battery with optional Intel® Next-Gen Wireless N technology enable 2x3 Draft N implementations with 2 spatial streams. Actual res may vary based on your specific hardware, connection rate, conditions and software configurations. See http://www.intel.c performance/mobile/wireless/index.htm for more informat Check with your PC and access point manufacturer for details.

- Processor complexity is good enough
- Transistor sizes can still scale
- Slow processors down to manage power
- Get performance from…

Parallelism



Top 5 Must-Haves

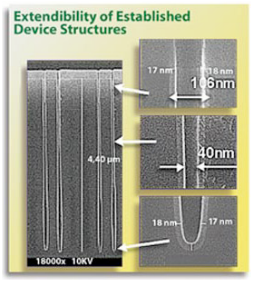
☐ **POWERFUL PROCESSOR**
A portrait of performance. "My generative portraits are demanding on the processors in my laptop, as they continuously manipulate video," says Lincoln. Thankfully, the **dual-core performance** of Intel Centrino processor technology can handle intensive tasks with flying colors.

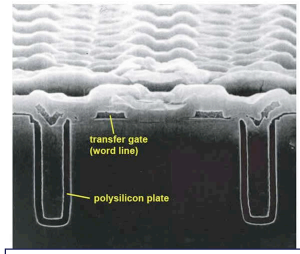
Consider:

- A chip with 1 processor core that takes 1 ns to perform an operation
- A chip with 2 processor cores – each of which takes 2 ns to perform an operation

Oh … transistors used for memory too

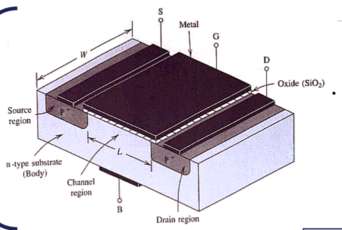
DRAM transistors are "deep trenches"

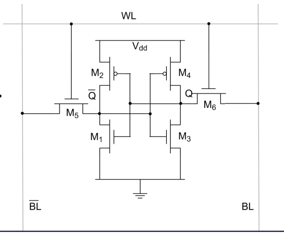




1 transistor (dense) memory

SRAM transistors made with logic process





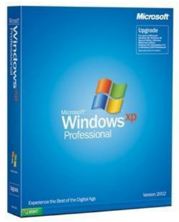
6 transistor (fast) memory

http://web.sfc.keio.ac.jp/~rdv/keio/sfc/teaching/architecture/architecture-2008/6t-SRAM-cell.png
http://www.micromagazine.com/archive/02/06/0206MI22b.jpg
http://www.ieee.org/portal/site/sscs/menuitem.f07ee9e3b2a01d06bb9305765bac26c8/index.jsp?&pName=sscs_level1_article&TheCat=2171&path=sscs/08Winter&file=Sunami.xml

Oh … transistors used for memory too

SRAM memory forms an *on-chip cache*

Problem: Program, data = bigger + power problem isn't going away…



- Why? Put faster memory closer to processing logic…
  - SRAM: avg. improvement: density +25%, speed +20%
  - DRAM: avg. improvement: density +60%, speed +4%
  - DISK: avg. improvement: density +25%, speed +4%

Leads to a "memory wall" – can't get data to processor fast enough

"Scaling the memory wall…"

Create a memory hierarchy…

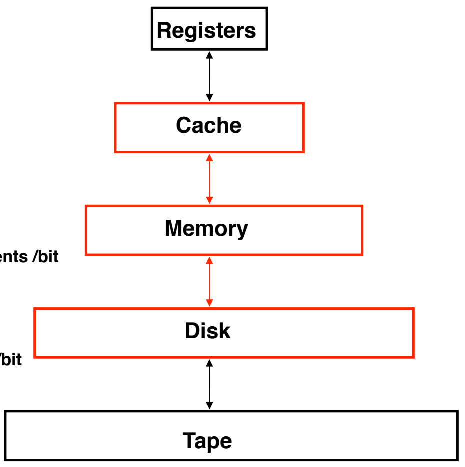
CPU Registers
100s Bytes
<10s ns

Cache
K Bytes
10-100 ns
1-0.1 cents/bit

Main Memory
M Bytes
200ns- 500ns
$.0001-.00001 cents /bit

Disk
G Bytes, 10 ms
(10,000,000 ns)
10^{-5} - 10^{-6} cents/bit

Tape
infinite
sec-min
10^{-8}



Put frequently used instructions and data in "fast" memory

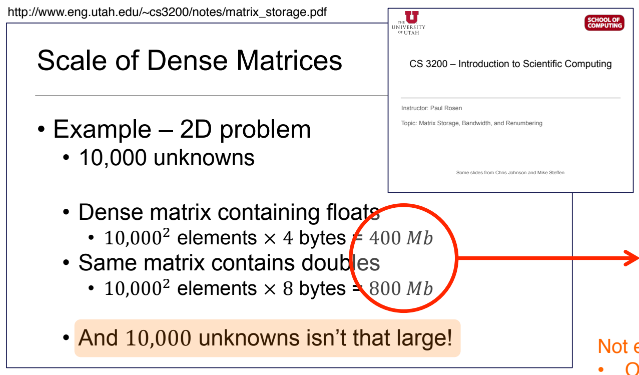
Have larger memory, storage options for all, less frequently used information

The fallout…

- Just a few examples that highlight the importance of understanding a computer's architecture … whether you're interested in HW or SW
  - (many more throughout the semester)

Example 1: the “memory wall”

http://www.eng.utah.edu/~cs3200/notes/matrix_storage.pdf



THE UNIVERSITY OF UTAH

SCHOOL OF COMPUTING

CS 3200 – Introduction to Scientific Computing

Instructor: Paul Rosen

Topic: Matrix Storage, Bandwidth, and Renumbering

Some slides from Chris Johnson and Mike Steffen

Scale of Dense Matrices

- Example – 2D problem
  - 10,000 unknowns
  - Dense matrix containing floats
    - $10{,}000^2$ elements $\times$ 4 bytes = $400\ Mb$
  - Same matrix contains doubles
    - $10{,}000^2$ elements $\times$ 8 bytes = $800\ Mb$
  - And 10,000 unknowns isn’t that large!

So, can this amount of data fit in fast, on-chip memory?

Not even close. 14 MB
- Optimizations can help, but still problematic.

Model	Cores	L1 Cache	L2 Cache	L3 Cache	Release Date	Release Price
i7 – 970	6	6 x 32 KB	6 x 256 KB	12 MB	07/19/10	$885
i7 – 980	6	6 x 32 KB	6 x 256 KB	12 MB	06/26/11	$583
i7 – 980X	6	6 x 32 KB	6 x 256 KB	12 MB	03/16/10	$999
i7 – 990X	6	6 x 32 KB	6 x 256 KB	12 MB	02/14/11	$999

Example 1: the “memory wall”

$$A=\begin{bmatrix} a & b \\ c & d \end{bmatrix}, B=\begin{bmatrix} e & f \\ g & h \end{bmatrix} \text{ then } AB=\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} e & f \\ g & h \end{bmatrix}=\begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

Need entire row, column of each per element.

Need entire row, columns to compute new row.

Assume “standard” matrix multiply code…

- Ideally, **(i)** 1 row of 10,000 elements AND **(ii)** a $10{,}000^2$ *matrix* is in fast memory

```
for(i=0; i<XSIZE; i=i+1) {
    for(j=0; j<YSIZE; j=j+1) {
      r = 0;
      for(k=0; k<XSIZE; k=k+1) {
        r = r + y[i][k] * z[k][j];
      }
      x[i][j] = r;
    }
}
```

(can apply more efficient storage techniques, but does not eliminate the problem)

Example 1: the “memory wall”

- **Changing how you write your code can – dramatically – improve execution time…**
- **Consider:**

v1
```
for(i=0; i<XSIZE; i=i+1) {
    for(j=0; j<YSIZE; j=j+1) {
      r = 0;
      for(k=0; k<XSIZE; k=k+1) {
        r = r + y[i][k] * z[k][j];
      }
      x[i][j] = r;
    }
}
```

Vs.

v2
```
for(jj=0; jj<XSIZE; jj=jj+B)
  for(kk=0; kk<YSIZE; kk=kk+B)
  for(i=0; i<XSIZE; i=i+1)
    for(j=jj; j<min(jj+B,YSIZE); j=j+1) {
      r=0;
      for(k=kk; k<min(kk+B,XSIZE); k=k+1)
        r = r + y[i][k]*z[k][j];
      x[i][j] = x[i][j] + r;
    }
```

```
for i=0; i<5; i++ {
    a = (a*b) + c;
}
```

MULT temp,a,b # temp ← a*b
MULT r1,r2,r3 # r1 ← r2*r3

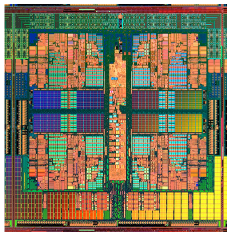
v2 requires 17% MORE instructions than v1, but takes 210% less time to run than v1!

Example 1: the memory wall

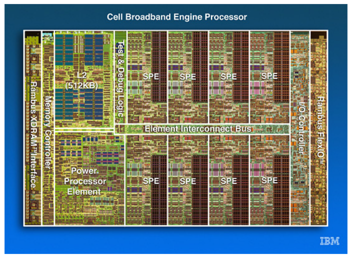
- **You’ll learn why a minor code re-write could lead to 2010% performance improvements this semester**
  - **(and this this is just for 1000 x 1000 matrices!)**
- **Foreshadowing…**
  - **An understanding of the underlying HW is essential**

Example 2: multi-core extensions…

Quad core chips...



7, 8, and 9 core chips...



When will it stop?

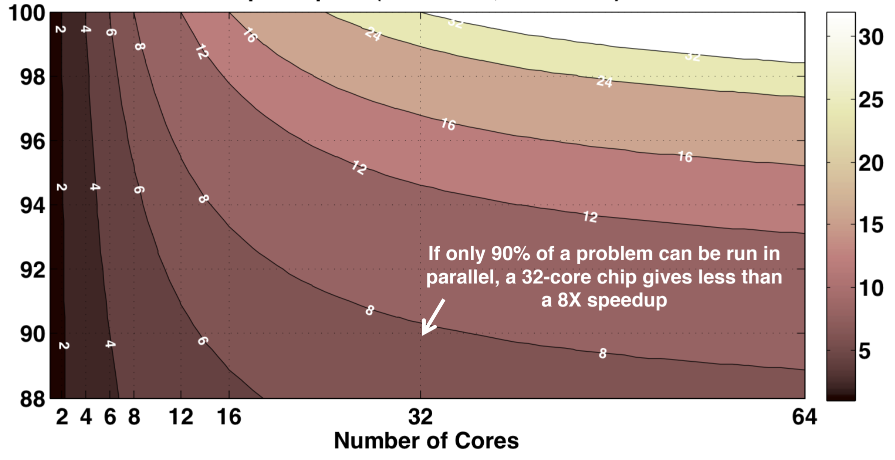
Practical problems must be addressed!

Multi-core only as good as its algorithms

Performance improvement defined by Amdhahl's Law

$$\text{Speedup} = \frac{1}{\left[1 - \text{Fraction}_{\text{parallelizable}}\right] + \dfrac{\text{Fraction}_{\text{parallelizable}}}{N}}$$

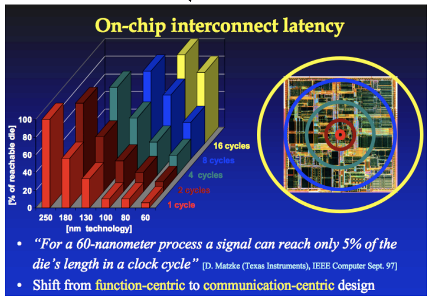
Speedup vs. (# of Cores, % Parallel)



Other issues…

- You may write nice parallel code, but implementation issues must also be considered
  - Example:
    - Assume N cores solve a problem in parallel
    - At times, N cores need to communicate with one another to continue computation
    - Overhead is NOT parallelizable...and can be significant!

$$\text{Speedup} = \frac{1}{\left[1 - \text{Fraction}_{\text{parallelizable}}\right] + \dfrac{\text{Fraction}_{\text{parallelizable}}}{N}}$$



On-chip interconnect latency

- *"For a 60-nanometer process a signal can reach only 5% of the die's length in a clock cycle"* [D. Matzke (Texas Instruments), IEEE Computer Sept. 97]
- Shift from function-centric to communication-centric design

Other issues…

- What if you write well-optimized, debugged code for a 4-core processor … which is then replaced with an 8-core processor?
  - Does code need to be re-written, re-optimized?
  - How do you adapt?

Example #3: technology's changing…

An Energy-Efficient Heterogeneous CMP based on Hybrid TFET-CMOS Cores

Vinay Saripalli[†], Asit K. Mishra[†], Suman Datta[§] and Vijaykrishnan Narayanan[†]
The Pennsylvania State University, University Park, PA 16802, USA
[†]{vxs924,amishra,vijay}@cse.psu.edu, [§]sdatta@engr.psu.edu

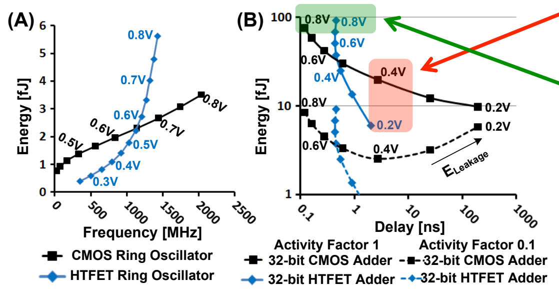


Figure 6: Energy-Delay performance comparision for (A) a CMOS and a HTFET ring-oscillator and (B) a CMOS 32-bit and a TFET 32-bit Adder.

You need to know what HW you have!

New transistors can affect trends discussed earlier.

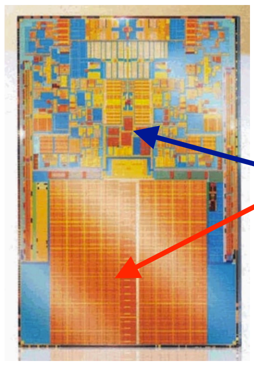
- HW that adds 2 numbers together can be more energy efficient with *lower* V for the same delay
- But if high performance *really* needed, older devices are better
- Future outlook?
  - Heterogeneous cores?
  - If task critical, use CMOS, if not, use TFET
- What about things like barrier synchronization?

Let's discuss some course goals (i.e. what you're going to learn)

Goal #1

- At the end of the semester, you should be able to...
  - **...describe the fundamental components required in a single core of a modern microprocessor**
    - (Also, explain how they interact with each other, with main memory, and with external storage media...)

Example



How do on-chip memory, processor logic, main memory, disk interact?

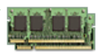
2.0 GB

$200.00

Apple Memory Module 2GB 667MHz DDR2 (PC2-5300) 2x1GB SO-DIMMs

Estimated Ship: Within 24 hours

Free Shipping



750GB SATA Hard Disk Drive Kit for...

Ships: Within 24hrs

Free Shipping

★★★★★

$299.00

Goal #2: (example: which plane is best?)

Which is best?

Plane	People	Range (miles)	Speed (mph)	Avg. Cost (millions)
737-800	162	3,060	530	63.5
747-8I	467	8000	633	257.5
777-300	368	5995	622	222
787-8	230	8000	630	153

Goal #2

- At the end of the semester, you should be able to…
  - …compare and contrast different computer architectures to determine which one performs better…

Example

AMD Smarter Choice

Processor	AMD Athlon™
Model	3200+
OPN Tray	ADA3200AEP5AR
OPN PIB	ADA3200BOX
Operating Mode 32 Bit	Yes
Operating Mode 64 Bit	Yes
Revision	CG
Core Speed (MHz)	2000
Voltages	1.50V

Intel® Pentium® Dual-Core processor

The Intel® Pentium® dual-core processor delivers great performance, low everyday computing.

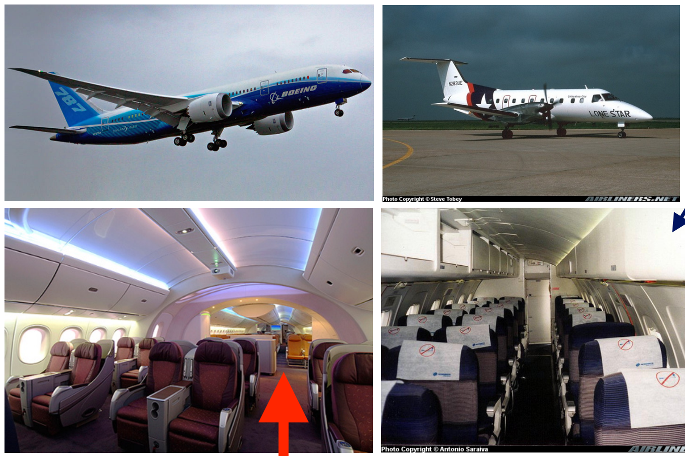
» Learn more

Processor Number¹	Architecture	Cache	Clock Speed	Front Side Bus	Dual-core
E2220	65 nm	1MB L2	2.40 GHz	800 MHz	✓
E2200	65 nm	1MB L2	2.20 GHz	800 MHz	✓
E2180	65 nm	1MB L2	2.00 GHz	800 MHz	✓
E2160	65 nm	1MB L2	1.80 GHz	800 MHz	✓
E2140	65 nm	1MB L2	1.60 GHz	800 MHz	✓

If you want to do X, which processor is best?

Goal #3: (motivation)

- Which plane would you rather fly to South Bend?


787
BOEING
LONE STAR
Photo Copyright © Steve Tobey
AIRLINERS.NET
Photo Copyright © Antonio Saraiva
AIRLINERS.NET

But this is the one that you (uncomfortably) do fly…

Photo Credits:
http://www.airliners.net/aircraft-data/stats.main?id=196
http://images.businessweek.com/ss/07/06/0615_boeing787/image/787interior.jpg

Goal #3

- At the end of the semester, you should be able to…
  - …apply knowledge about a processor's datapath, different memory hierarchies, performance metrics, etc. to design a microprocessor that (a) meets a target set of performance goals and (b) is realistically implementable

Example

Climate Agency Awards $350 Million For Supercomputers

The National Oceanic and Atmospheric Administration will pay CSC and Cray to plan, build, and operate high-performance computers for climate prediction research.

By J. Nicholas Hoover
InformationWeek

<6 MHz MP3 Decode

Tensilica optimized the MP3 decoder for its HiFi DSPs. This MP3 decoder now runs at the lowest power and is the most efficient in the industry, requiring just 5.7 MHz when running at 128 Kbps, 44.1 KHz and dissipating 0.45 mW in TSMC's 65nm LP process (including memories). This makes Tensilica's Xtensa HiFi 2 Audio Engine ideal for adding MP3 playback to cellular phones, where current carrier requirements are for 100 hours of playback time on a battery charge, and increasing to 200 hours in the near future.

http://www.tensilica.com/products/audio/low-power-mp3-playback.htm

(click for larger image and for full photo gallery)

In terms of a research and development computer, NOAA found it requires one the power of which will be ultimately measured in petaflops, which would make the future machine one of the world's most powerful supercomputers.

http://www.informationweek.com/news/government/enterprise-apps/showArticle.jhtml?articleID=225000152

Goal #4

- At the end of the semester, you should be able to…
  - …understand how code written in a high-level language (e.g. C) is eventually executed on-chip…

Example

In C:

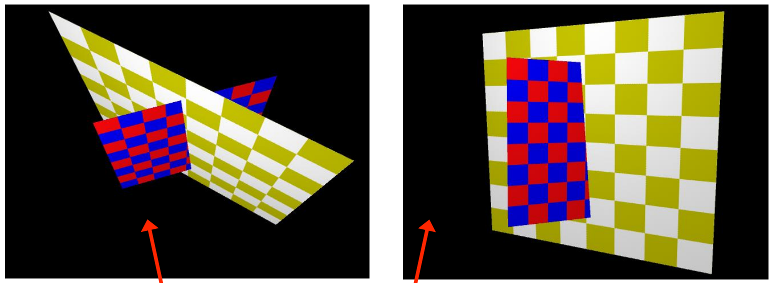
```
void insertionSort(int numbers[], int array_size)
{
  int i, j, index;

  for (i=1; i < array_size; i++)
  {
    index = numbers[i];
    j = i;
    while ((j > 0) && (numbers[j-1] > index))
    {
      numbers[j] = numbers[j-1];
      j = j - 1;
    }
    numbers[j] = index;
  }
}
```

In Java:

```
public static void insertionSort(int[] list, int length) {
    int firstOutOfOrder, location, temp;

    for(firstOutOfOrder = 1; firstOutOfOrder < length; firstOutOfOrder++) {
        if(list[firstOutOfOrder] < list[firstOutOfOrder - 1]) {
            temp = list[firstOutOfOrder];
            location = firstOutOfOrder;

            do {
                list[location] = list[location-1];
                location--;
            }
            while (location > 0 && list[location-1] > temp);

            list[location] = temp;
        }
    }
}
```

Both programs could be run on the same processor…
How does this happen?

Goal #5: (motivation, part 1)



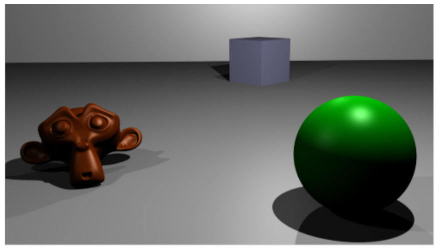
For this image, and this perspective, which pixels should be rendered?

Solution provided by *z-buffering algorithm*

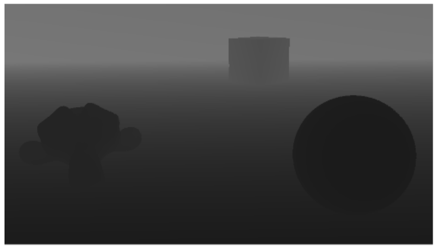
- Depth of each object in 3D scene used to paint 2D image
- Algorithm steps through list of polygons
  - # of polygons tends to be >> (for more detailed scene)
  - # of pixels/polygon tends to be small

Image source: www.cs.unc.edu/~pmerrell/comp575/Z-Buffering.ppt

Goal #5: (motivation, part 2)

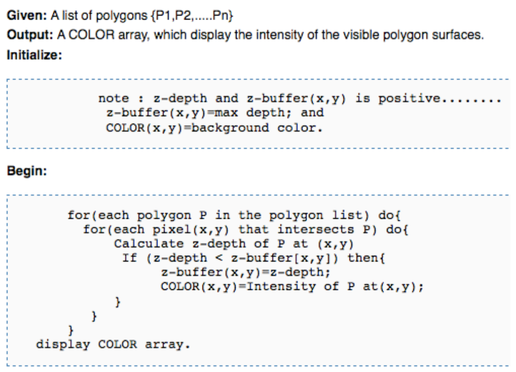


A simple three-dimensional scene



Z-buffer representation

Often a dynamic data structure (e.g. linked list)



Given: A list of polygons {P1,P2,.....Pn}
Output: A COLOR array, which display the intensity of the visible polygon surfaces.
Initialize:

```
        note : z-depth and z-buffer(x,y) is positive........
         z-buffer(x,y)=max depth; and
         COLOR(x,y)=background color.
```

Begin:

```
    for(each polygon P in the polygon list) do{
      for(each pixel(x,y) that intersects P) do{
          Calculate z-depth of P at (x,y)
           If (z-depth < z-buffer[x,y]) then{
                z-buffer(x,y)=z-depth;
                COLOR(x,y)=Intensity of P at(x,y);
           }
       }
     }
display COLOR array.
```

Image source: http://en.wikipedia.org/wiki/Z-buffering

Goal #5 Motivation (part 3)

Data structure may not fit in fast memory

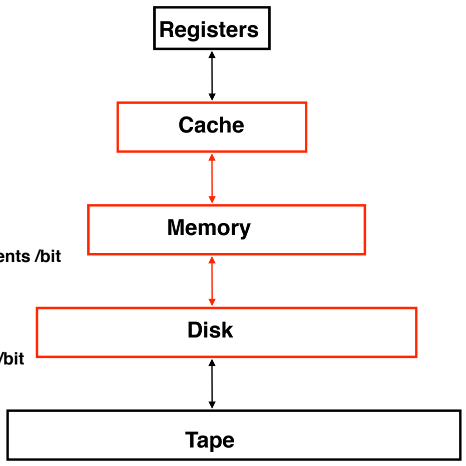
CPU Registers
100s Bytes
<10s ns

Cache
K Bytes
10-100 ns
1-0.1 cents/bit

Main Memory
M Bytes
200ns- 500ns
$.0001-.00001 cents /bit

Disk
G Bytes, 10 ms
(10,000,000 ns)
10^{-5} - 10^{-6} cents/bit

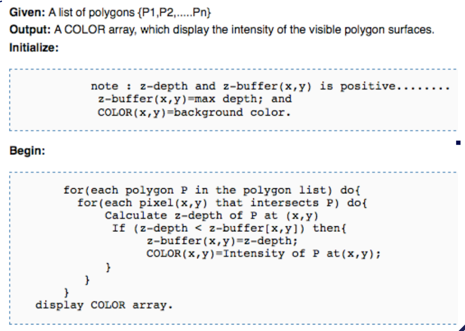
Tape
infinite
sec-min
10^{-8}



Goal #5

- At the end of the semester, you should be able to...
  - ...use knowledge about a microprocessors underlying hardware (or “architecture”) to write more efficient software…

Example



Given: A list of polygons {P1,P2,.....Pn}
Output: A COLOR array, which display the intensity of the visible polygon surfaces.
Initialize:

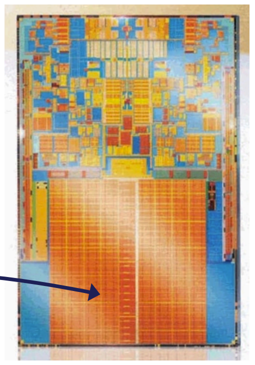
```
        note : z-depth and z-buffer(x,y) is positive........
         z-buffer(x,y)=max depth; and
         COLOR(x,y)=background color.
```

Begin:

```
    for(each polygon P in the polygon list) do{
      for(each pixel(x,y) that intersects P) do{
          Calculate z-depth of P at (x,y)
           If (z-depth < z-buffer[x,y]) then{
                z-buffer(x,y)=z-depth;
                COLOR(x,y)=Intensity of P at(x,y);
           }
       }
     }
display COLOR array.
```

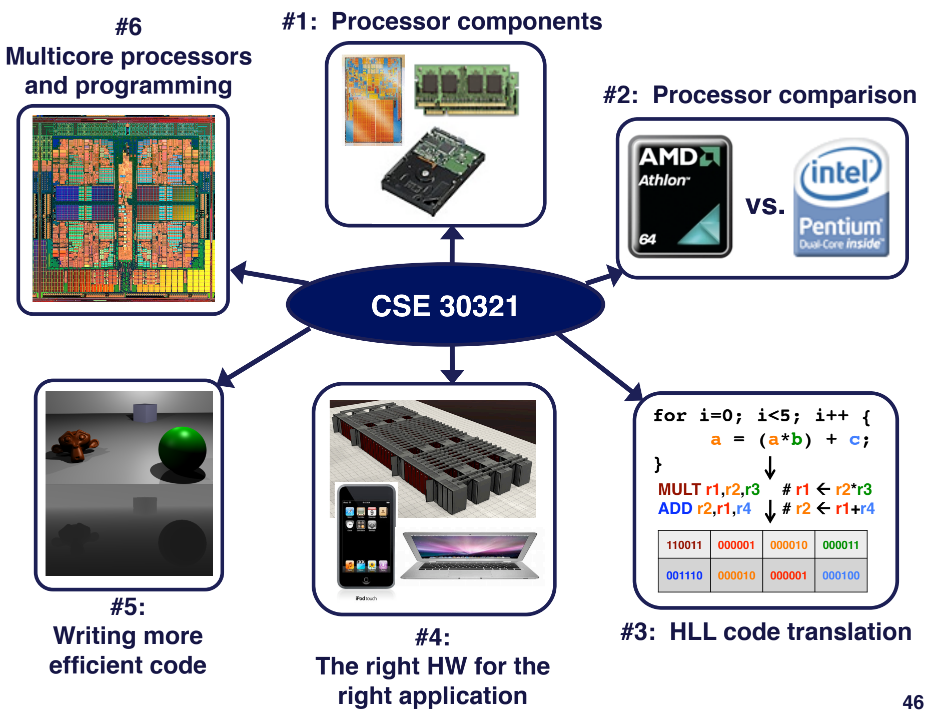
Write this code…

…so it always fits in fast, on-chip memory



Goal #6

- At the end of the semester, you should be able to…
  - …explain and articulate why modern microprocessors now have more than 1 core and how SW must adapt to accommodate the now prevalent multi-core approach to computing
- Why?
  - For 8, 16 core chips to be practical, we have to be able to *use them*


CSE 30321
#6 Multicore processors and programming
#1: Processor components
#2: Processor comparison
AMD Athlon 64
vs.
intel Pentium Dual-Core inside
#3: HLL code translation
#4: The right HW for the right application
#5: Writing more efficient code

```
for i=0; i<5; i++ {
        a = (a*b) + c;
}
MULT r1,r2,r3    # r1 ← r2*r3
ADD r2,r1,r4     # r2 ← r1+r4
```

110011	000001	000010	000011
001110	000010	000001	000100