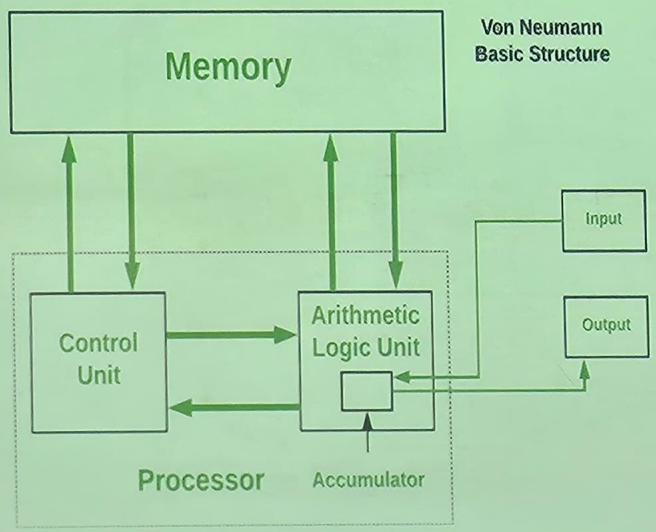
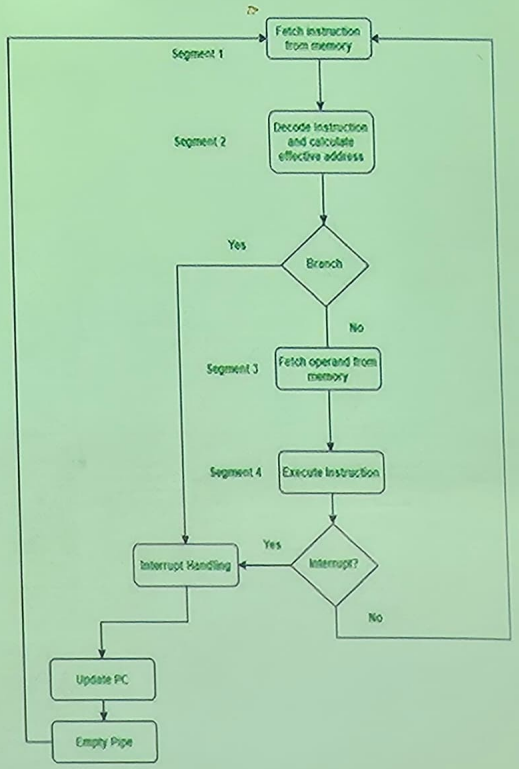
Processor Basics

What is a Processor?

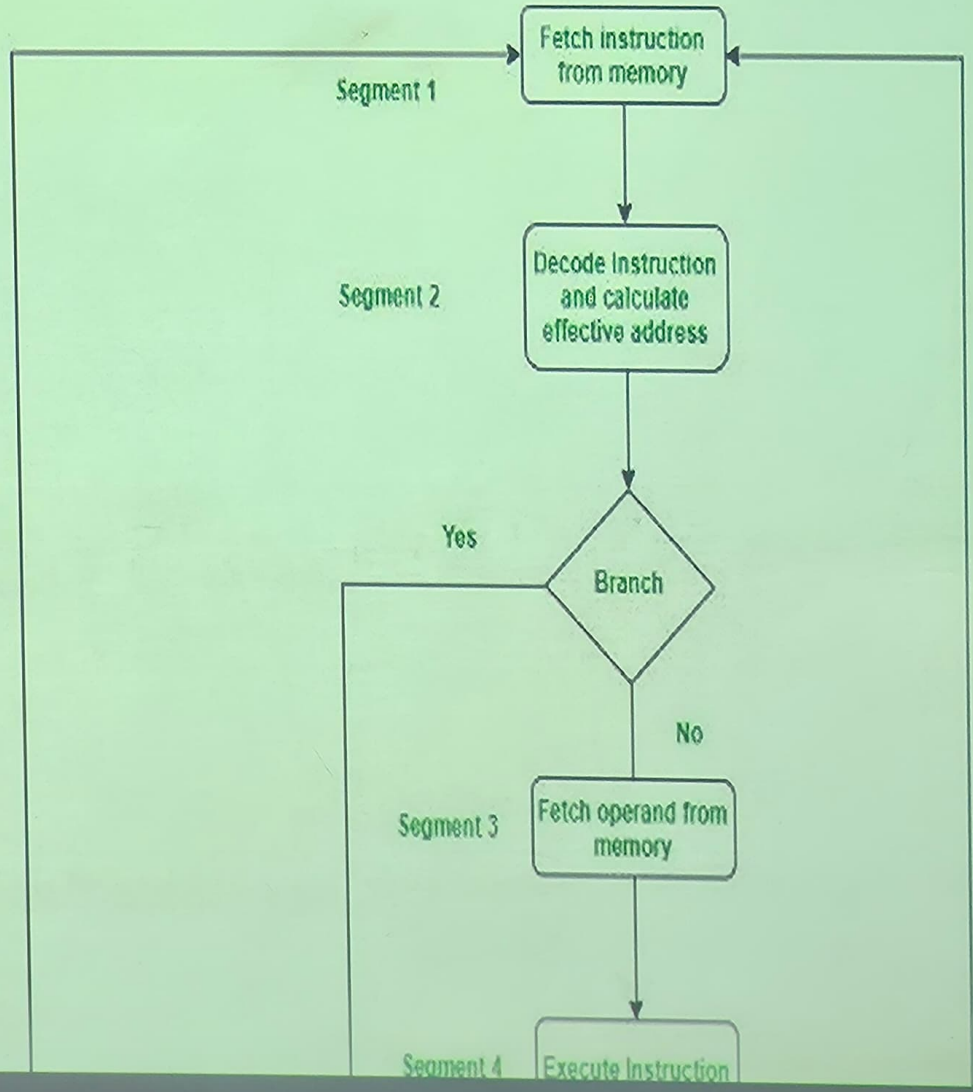
- **CPU** = executes instructions
- Coordinates data movement and control



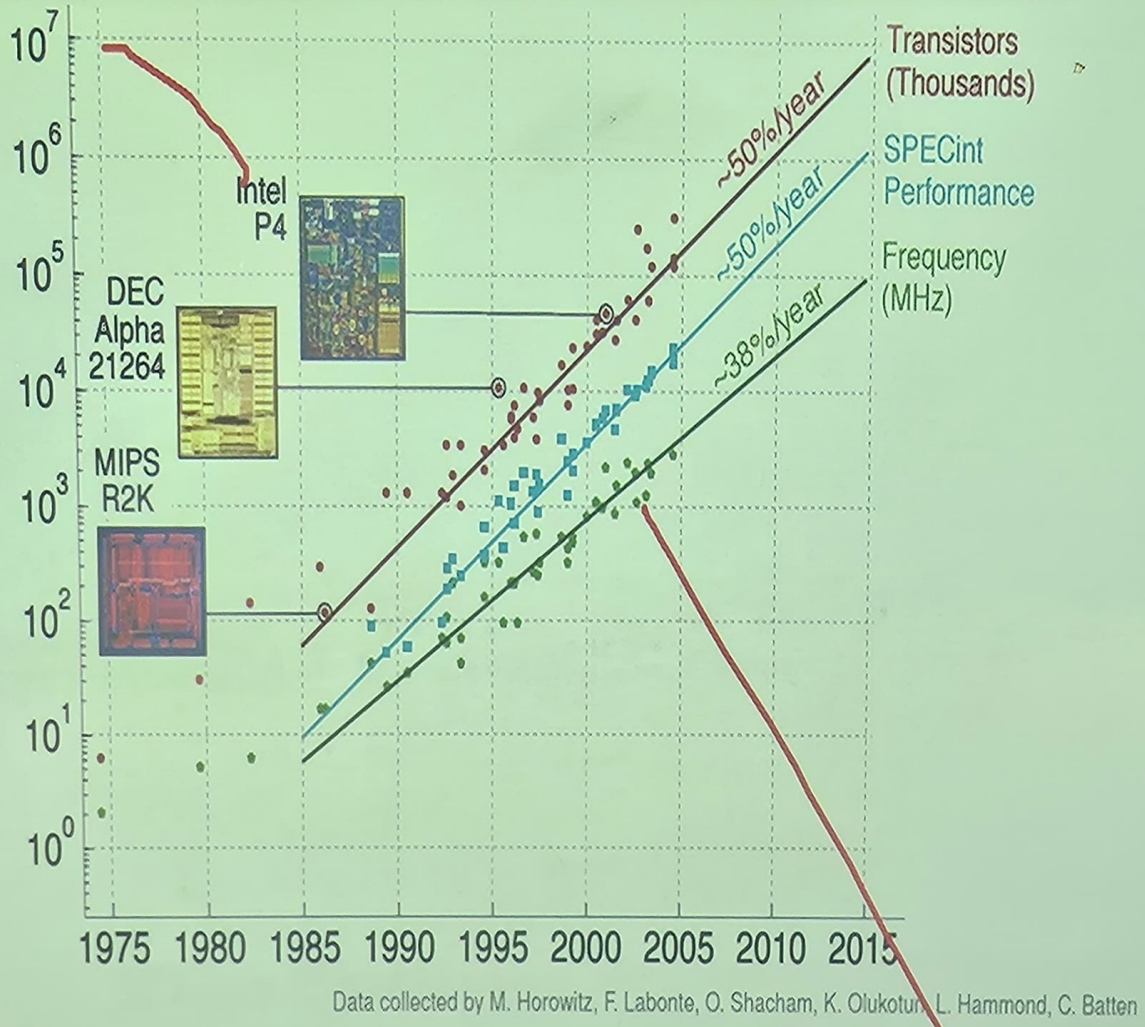


or Basics

ntrol

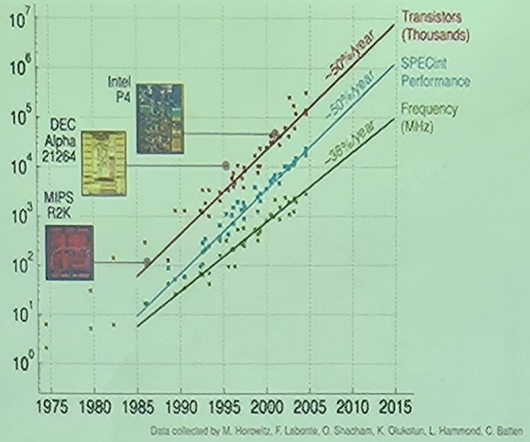


Scaling: Processor,

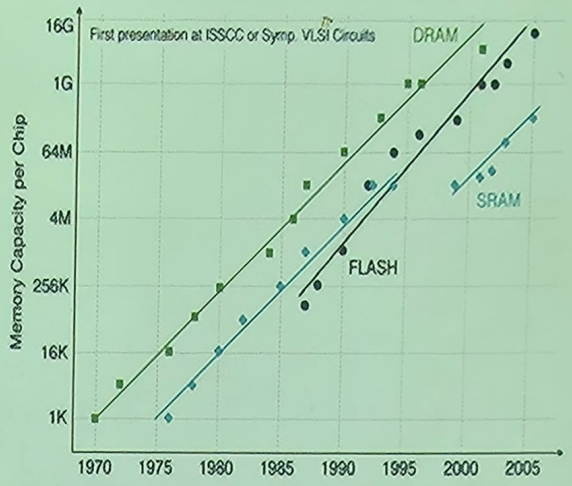


Data collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, C. Batten

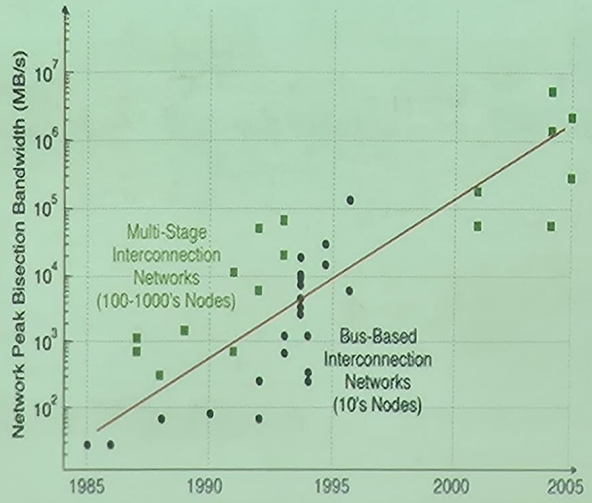
Scaling: Processor, Memory and Network



Data collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, C. Batten

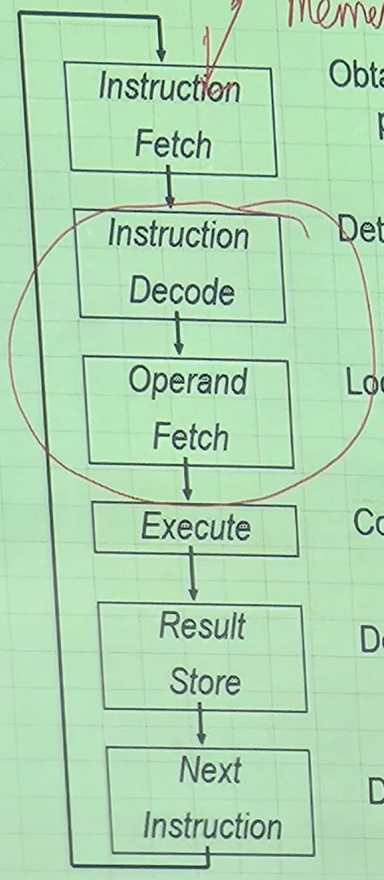


Adapted from K. Itoh et al, "Ultra-Low Voltage Nano-Scale Memories," Spring 2007.



Data from Hennessy & Patterson, Morgan Kaufmann, 2nd & 5th eds., 1996 & 2011; D.E. Culler et al., Morgan Kaufmann, 1999.

Fundamental Execution Cycle



Memory is slower than processor | word

Obtain instruction from program storage

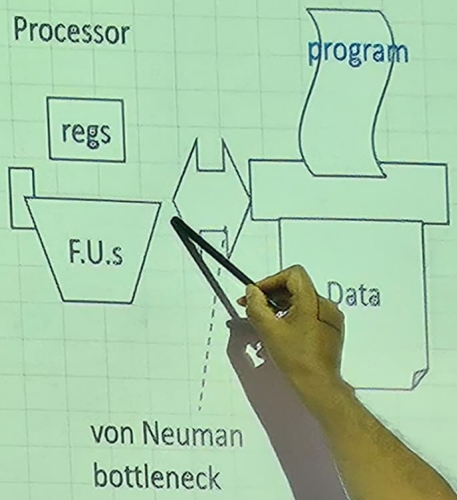
Determine required actions and instruction size

Locate and obtain operand data

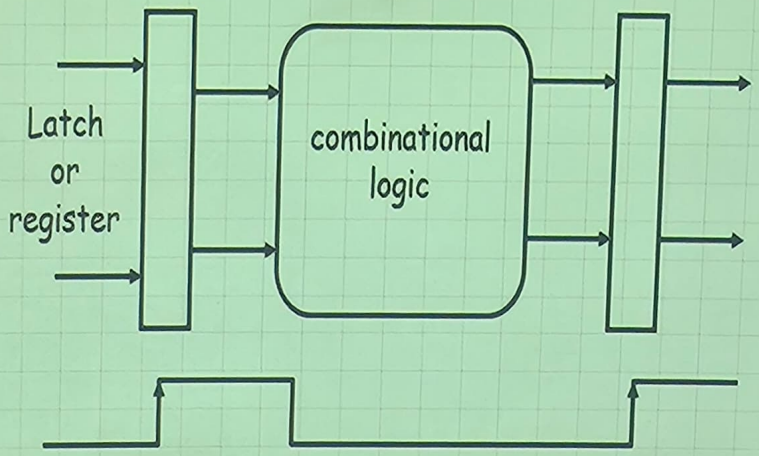
Compute result value or status

Deposit results in storage for later use

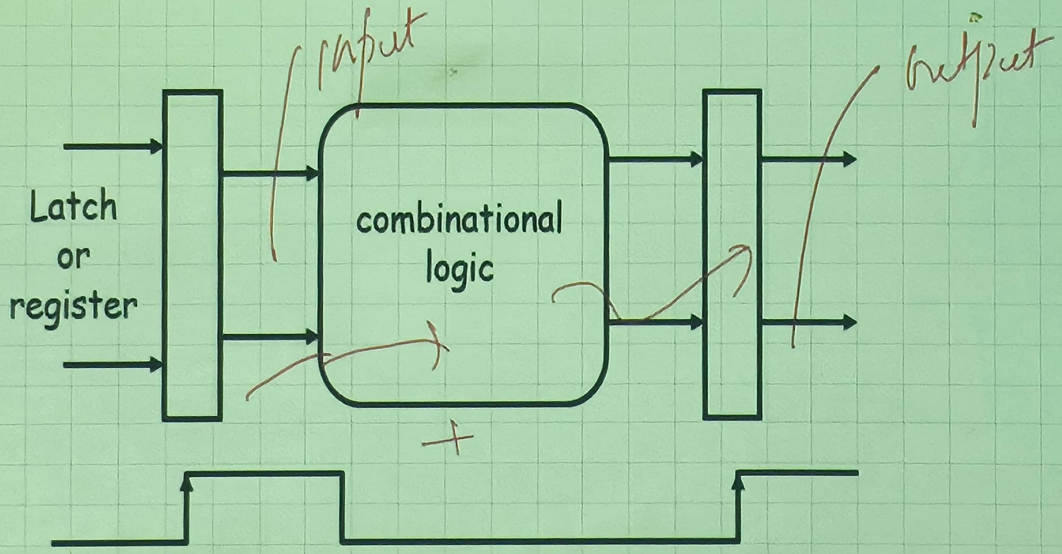
Determine successor instruction



What's a Clock Cycle?



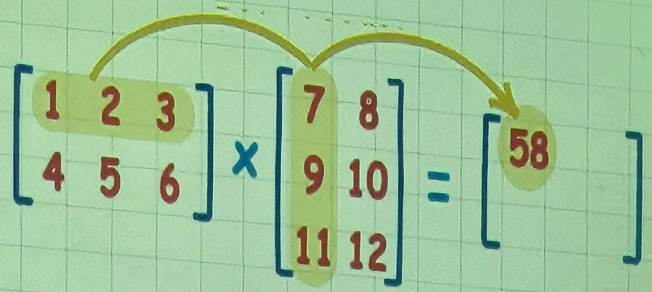
What's a Clock Cycle?



- 10 levels of gates
  - clock propagation, wire lengths, drivers

Matrix Multiplication

```
for i in xrange(4096):
  for j in xrange(4096):
    for k in xrange(4096):
      C[i][j] += A[i][k] *
B[k][j]
```

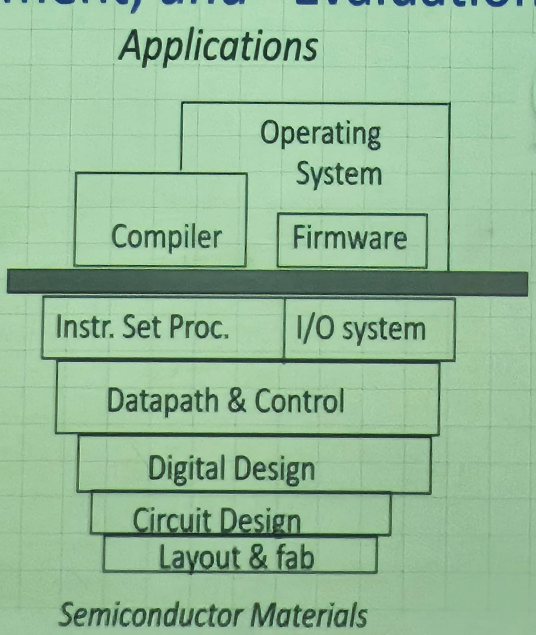


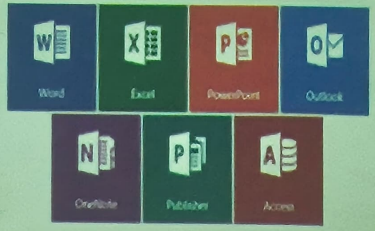
Implementation	Running time (s)	Absolute speedup
Python	25,552.48	1x
Java	2,372.68	11x
C	542.67	47x
Parallel loops	69.80	366x
Parallel divide and conquer	3.80	6,727x
plus vectorization	1.10	23,224x
plus AVX intrinsics	0.41	62,806x

Leiserson+, "There's plenty of room at the Top: What will drive computer performance after Moore's law?", Science, 2020

What is "Computer Architecture"?

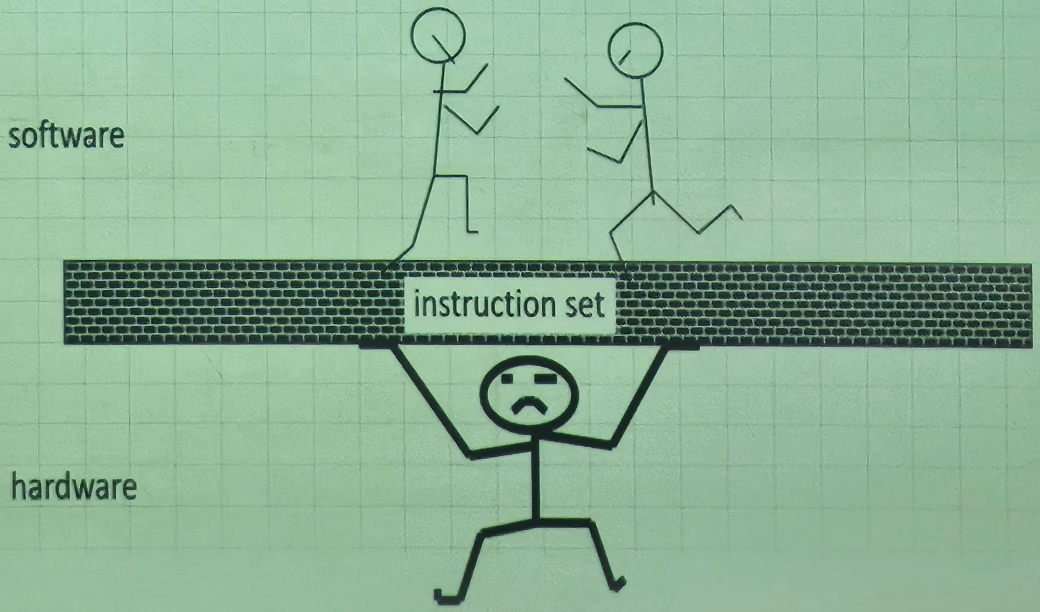
- Coordination of many *levels of abstraction*
- Under a rapidly *changing set of forces*
- Design, Measurement, *and* Evaluation





The Instruction Set: a Critical Interface

- Properties of a good abstraction
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels



```c
#include <stdio.h>
#include <math.h>
int main(void) {
  //          printf("main is : %f\n", sqrt(5));
            printf("main is : %f\n" );
 //           printf("main is : %f\n", abs(5));
            return 0;
}
```

~

~

~

~

~

~

~

~

~

~

~

~

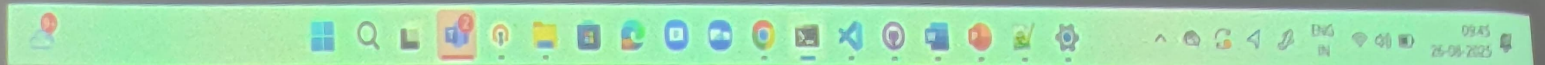
~

~

~

~

"helloWorld.c" 9L, 215B 4,35 All



```
      |                                %d
(base) kolin@mosaic:~/col7001$ ./a.out
main is : 2.236068
main is : 2.236068
(base) kolin@mosaic:~/col7001$ vi helloWorld.c

(base) kolin@mosaic:~/col7001$ gcc helloWorld.c
helloWorld.c: In function ‘main’:
helloWorld.c:5:32: warning: format ‘%f’ expects a matching ‘double’ argument [-Wformat=]
    5 |             printf("main is : %f\n" );
      |                               ~^
      |                                |
      |                                double
(base) kolin@mosaic:~/col7001$ gcc helloWorld.c
helloWorld.c: In function ‘main’:
helloWorld.c:5:32: warning: format ‘%f’ expects a matching ‘double’ argument [-Wformat=]
    5 |             printf("main is : %f\n" );
      |                               ~^
      |                                |
      |                                double
(base) kolin@mosaic:~/col7001$ ./a.out
main is : 0.000000
(base) kolin@mosaic:~/col7001$ vi helloWorld.c

(base) kolin@mosaic:~/col7001$ gcc helloWorld.c
```

```
      |                                  ~^
      |                                   |
      |                                  double
(base) kolin@mosaic:~/col7001$ gcc helloWorld.c
helloWorld.c: In function ‘main’:
helloWorld.c:5:32: warning: format ‘%f’ expects a matching ‘double’ argument [-Wformat=]
    5 |             printf("main is : %f\n" );
      |                                  ~^
      |                                   |
      |                                  double
(base) kolin@mosaic:~/col7001$ ./a.out
main is : 0.000000
(base) kolin@mosaic:~/col7001$ vi helloWorld.c

(base) kolin@mosaic:~/col7001$ gcc helloWorld.c
helloWorld.c: In function ‘main’:
helloWorld.c:5:32: warning: format ‘%f’ expects a matching ‘double’ argument [-Wformat=]
    5 |             printf("main is : %f\n" );
      |                                  ~^
      |                                   |
      |                                  double
(base) kolin@mosaic:~/col7001$ ./a.out
main is : 2.236068
main is : 2.236068
(base) kolin@mosaic:~/col7001$
```

A "Typical" RISC

- 32-bit fixed format instruction (3 formats)
- 32 32-bit GPR (R0 contains zero, DP take pair)
- 3-address, reg-reg arithmetic instruction
- Single address mode for load/store:
  base + displacement
  - no indirection
- Simple branch conditions
- Delayed branch

A "Typical" RISC

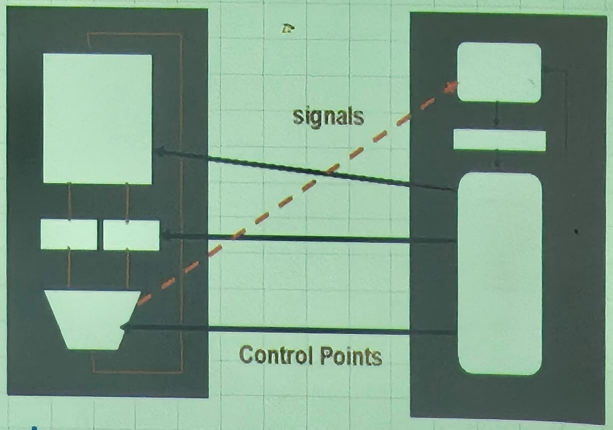
- 32-bit fixed format instruction (3 formats)
- 32 32-bit GPR (R0 contains zero, DP take pair)
- 3-address, reg-reg arithmetic instruction
- Single address mode for load/store:
  base + displacement
  - no indirection

see: SPARC, MIPS, HP PA-Risc, DEC Alpha, IBM PowerPC, CDC 6600, CDC 7600, Cray-1, Cray-2, Cray-3

- Simple branch conditions
- Delayed branch

Datapath Vs Control

- Datapath: Storage, FU, interconnect sufficient to perform the desired functions
  - Inputs are Control Points
  - Outputs are signals
- Controller: State machine to orchestrate operation on the data path
  - Based on desired function and signals



A Typical Datapath