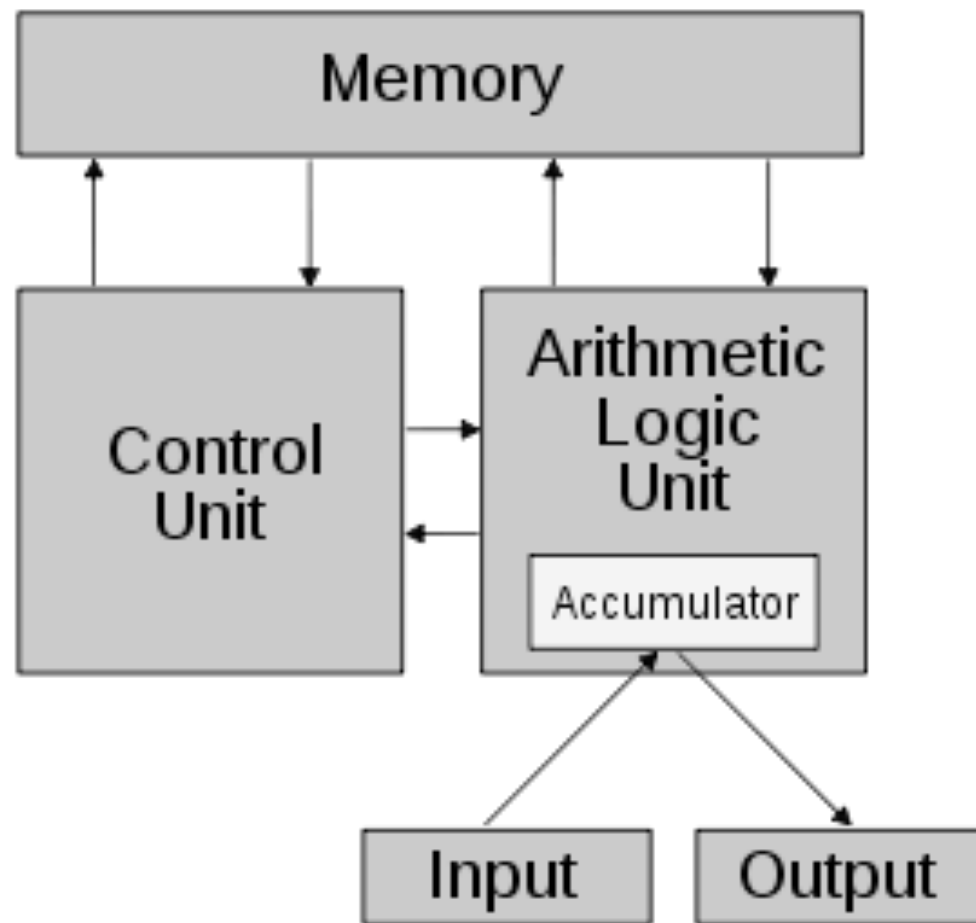


Computer architecture

(drastically simplified)

Von Neumann architecture



Program is stored
in memory

Wikipedia

Definitions (I)

- Memory (where we store things)
- Input/Output (read/write from external source)
- ALU- arithmetic logic unit (does the math)
- Control unit - Handle communication/issue tasks

The first computer

Z1-1938



Wikipedia

ZI

- Instructions read from punched tape
- Memory - 64 words
- Speed - 1 Hz
- Instruction set size : 8
- 1 to 20 clock cycles per instruction

Instruction set

- What operation should be performed on what data
- Example: add information in memory 2 and 3

Add R2 R3

000000 | 000000 | 0 000000 | 1

ZI (part I)

- Input/Output
 - Ask device for decimal number
 - Write to device a decimal number
- Memory
 - Read from memory to register 1 or 2
 - Write to memory from register 1 or 2

ZI (part 2)

- Arithmetic operations
 - Add floating point number R1 to R2
 - Subtract R1 from R2
 - Multiply R1 and R2
 - Divide R1 by R2
- Current intel architectures >1000 operations

Simple loop

a=0

for(i=0; i < 10; i++) a=a+i;

What this means

a=0

for(i=0; i < 10; i++) a=a+i;

1 a=0

2 i=0

10 if(i >= 10) goto 20

11 a=a+i

12 i=i+1

13 goto 10

20 .

What this means

a=0

for(i=0; i < 10; i++) a=a+i;

1 a=0

2 i=0

10 if(i >= 10) goto 20

11 a=a+i

12 i=i+1

13 goto 10

20 .

Copy 0 into register 1

Copy 0 into register 2

Copy 10 into register 3

Check R2<R3 11 20

Add register 2 3

Load 1 into register 3

Add register 2 and 3

Goto 10

Store a to memory

What this means

a=0

for(i=0; i < 10; i++) a=a+i;

1 a=0

2 i=0

10 if(i >= 10) goto 20

11 a=a+i

12 i=i+1

13 goto 10

20 .

Load instruction

Copy 0 into register 1

Load instruction

Copy 0 into register 2

Load instruction

Copy 10 into register 3

Load instruction

Check R2<R3 11 20

Load instruction

Add register 2 3

Load instruction

Load 1 into register 3

Load instruction

Add register 2 and 3

Goto 10

Store a to memory

Storage types

- Bit
 - 0 or 1
- A nibble
 - 4 bits (16 possible values)
- A byte
 - 8 bits (256 values)
 - Range 0-255 unsigned (or -127-128)

Storage types (part 2)

- Word
 - Often called a short
 - 2 bytes
 - 65536 values (0-65535) or (-32678-32677)
 - Characters are sometimes stored as word

Storage types (part 3)

- Dword
 - 4 bytes (32-bits)
 - Default storage mechanism for a integer and float in most languages (not python)
 - 32-bit memory storage mechanism

Dwords (integers)

- Range from a little more than < -2 billion to 2 billion
- Meant that computers couldn't access more than 4 GB
- Need to be careful in using an integer to access a very large array

Single precision floating point

- Representation
 - 1 sign, 8 exponential, 23 significant bits
 - ~Accuracy of 7.2 digits
- Problems adding floating point numbers
 - `float a=4000000,b=1;`
 - `a+=b`
 - `a` still equals 4000000
- Alternate fixed point

Endian problems

- How do you represent 200
 - 11001000 - big endian
 - 00010011 - little endian
- big endian - XDR
- little endian - Almost everything today

Double

- 8 bytes
- Integer
 - Up to 10^{19}
- float
 - 1 sign, 11 exponent, 52 significant
 - ~15.9 decimal points of accuracy

Quad

- 16 bytes (64 bits)
- Integers --- get serious
- Floats
 - 1 sign, 15 exponents, 112 significant
 - 34 digits of accuracy

Speed

- How many clock cycles to perform a given instruction
- Same order of magnitude 1-20 on current architectures
- Clock speed
 - 1946 1 Hz
 - 2000 4 GHz

Latency

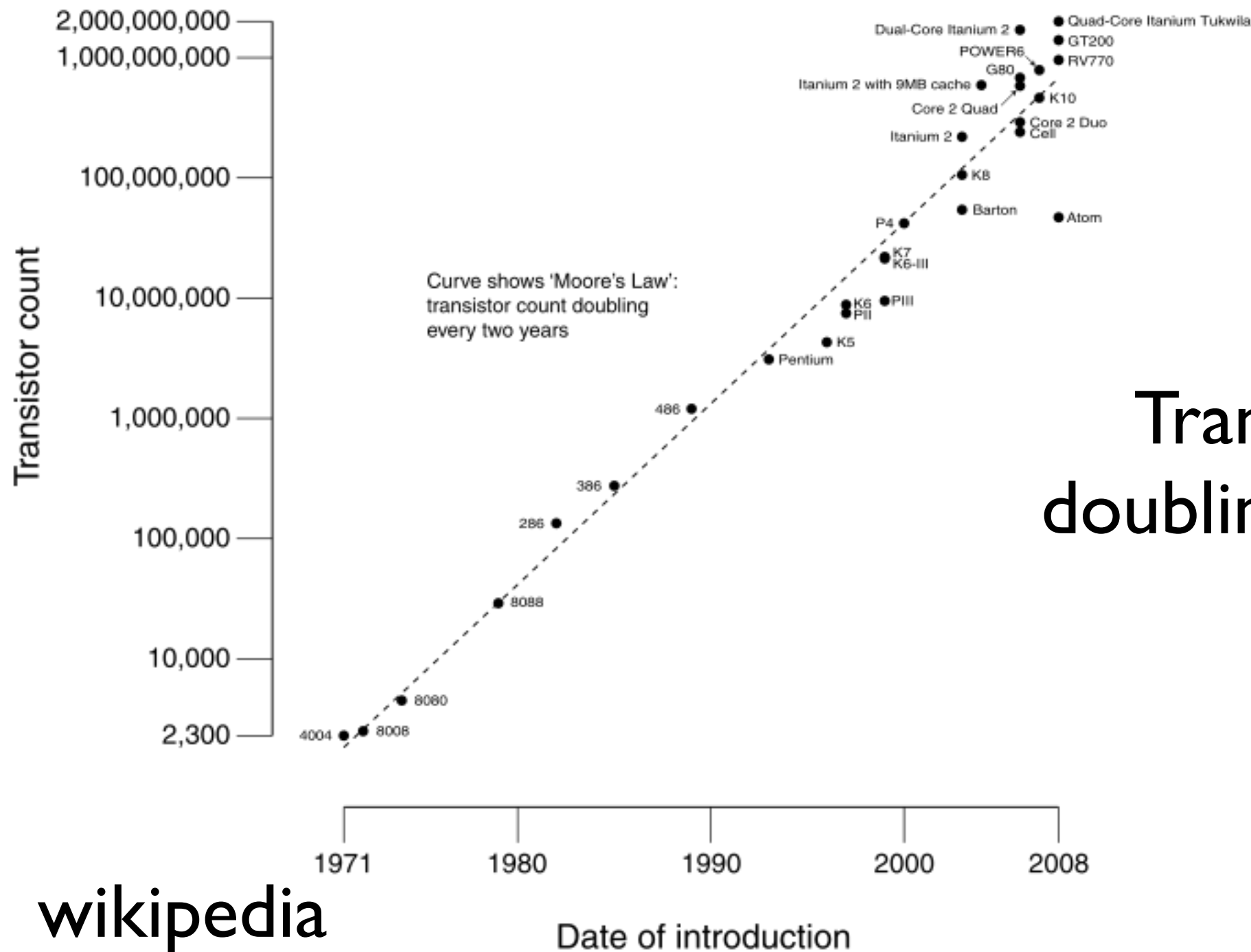
- Many operations take more than one clock cycle
- Particularly true in accessing memory
- The number of clock cycles (or time) it takes to retrieve a piece of information is referred to as the latency

Achieving performance on Von Neumann machines

- Limit the number of operations
- Limit the number of expensive operations
- Maximize reuse of registers
- Limit the calls to input/output devices

Moore's law

CPU Transistor Counts 1971-2008 & Moore's Law

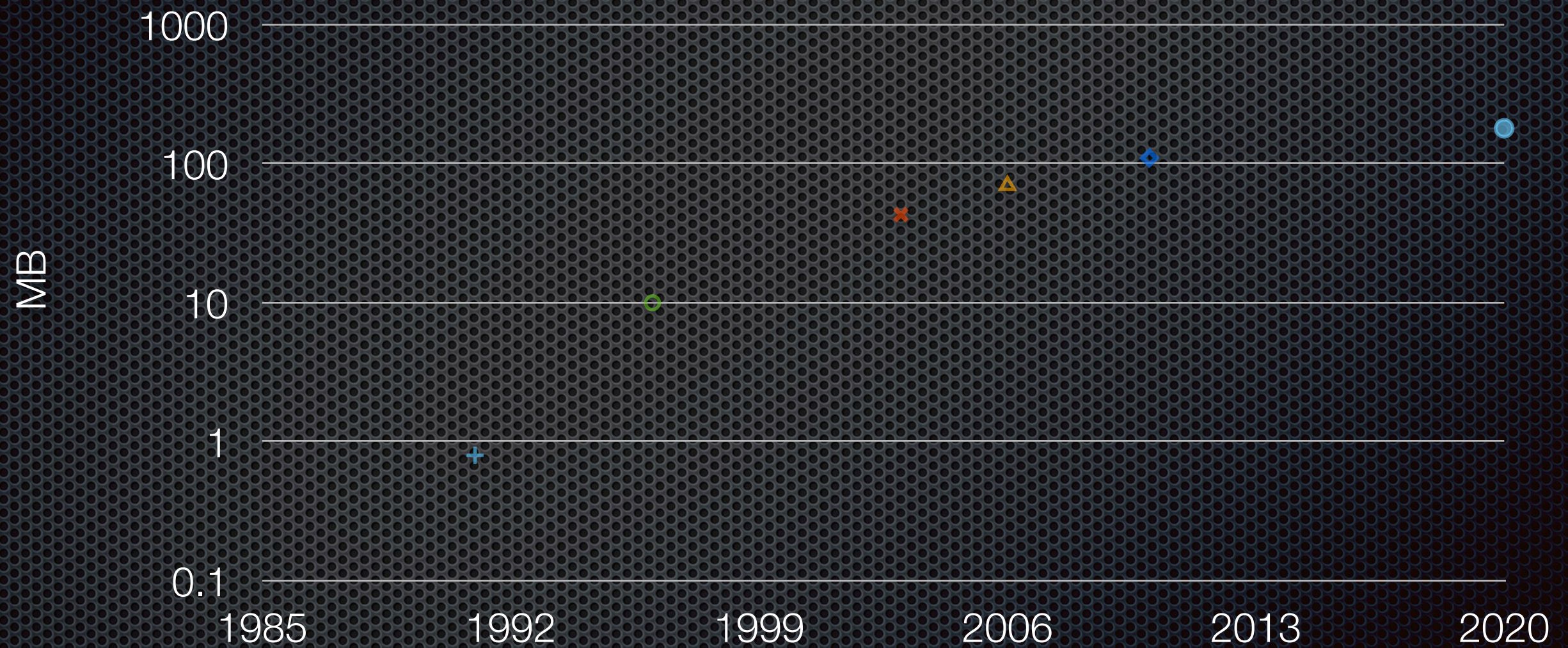


wikipedia

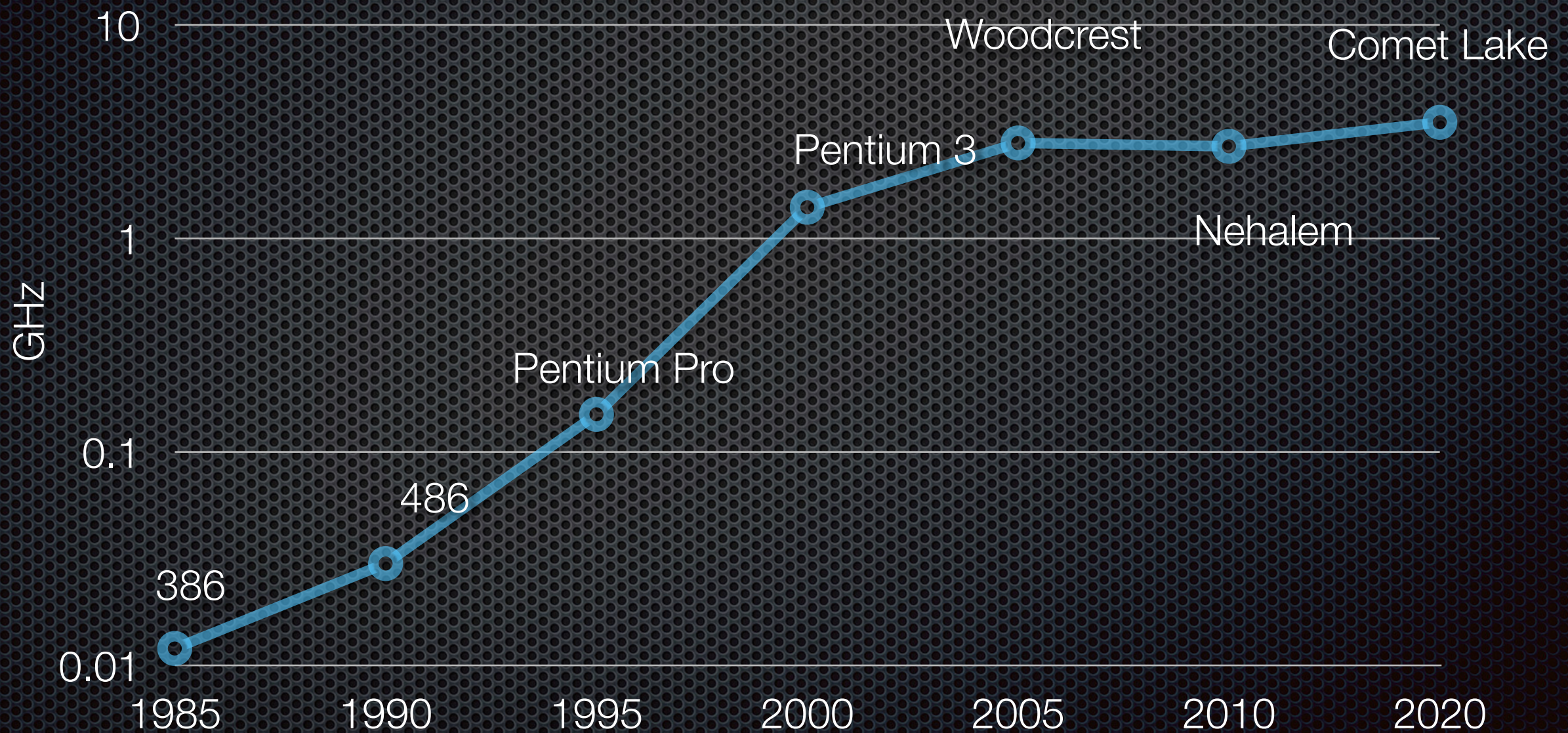
Disk size - Standard 3.5 inch PC disk



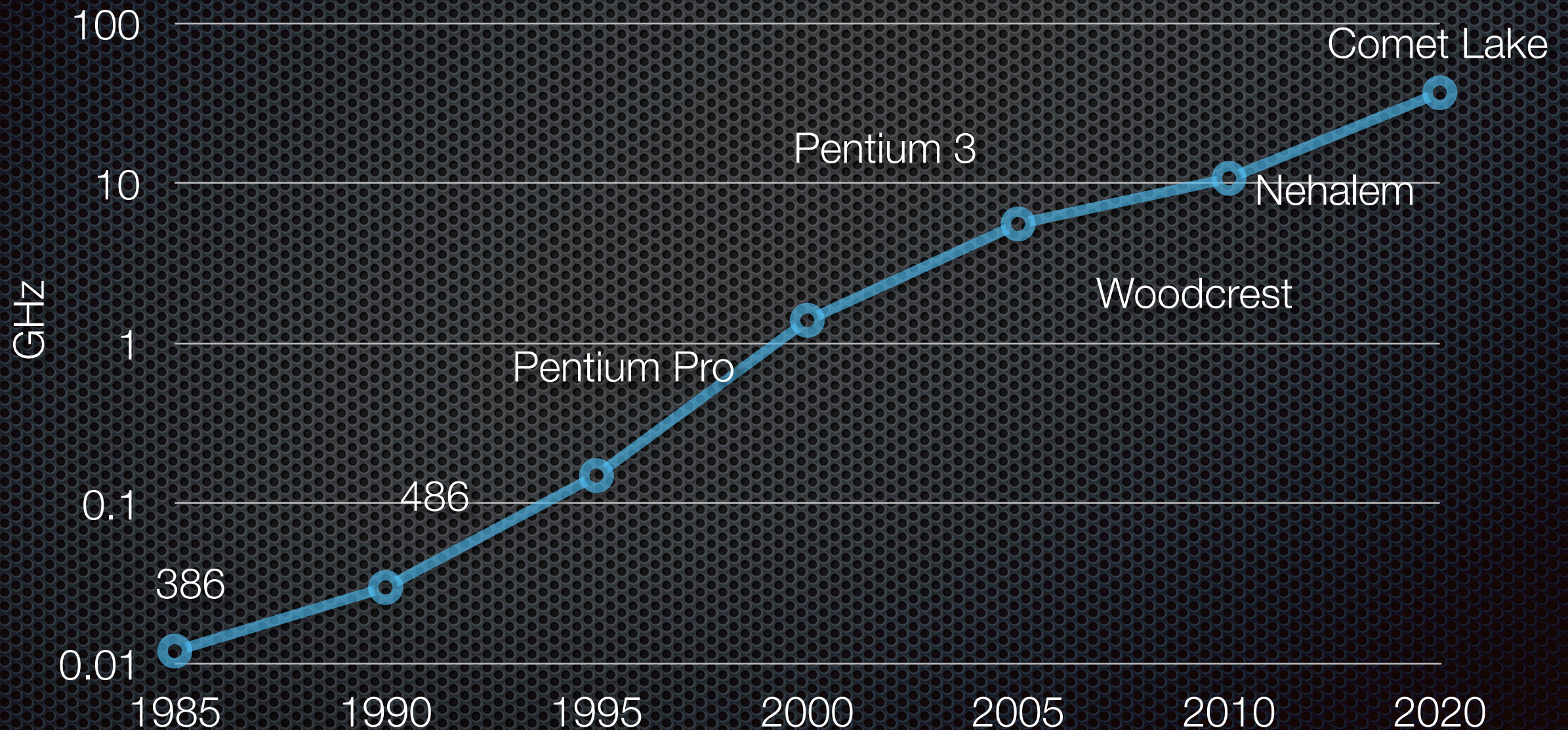
Disk Speed



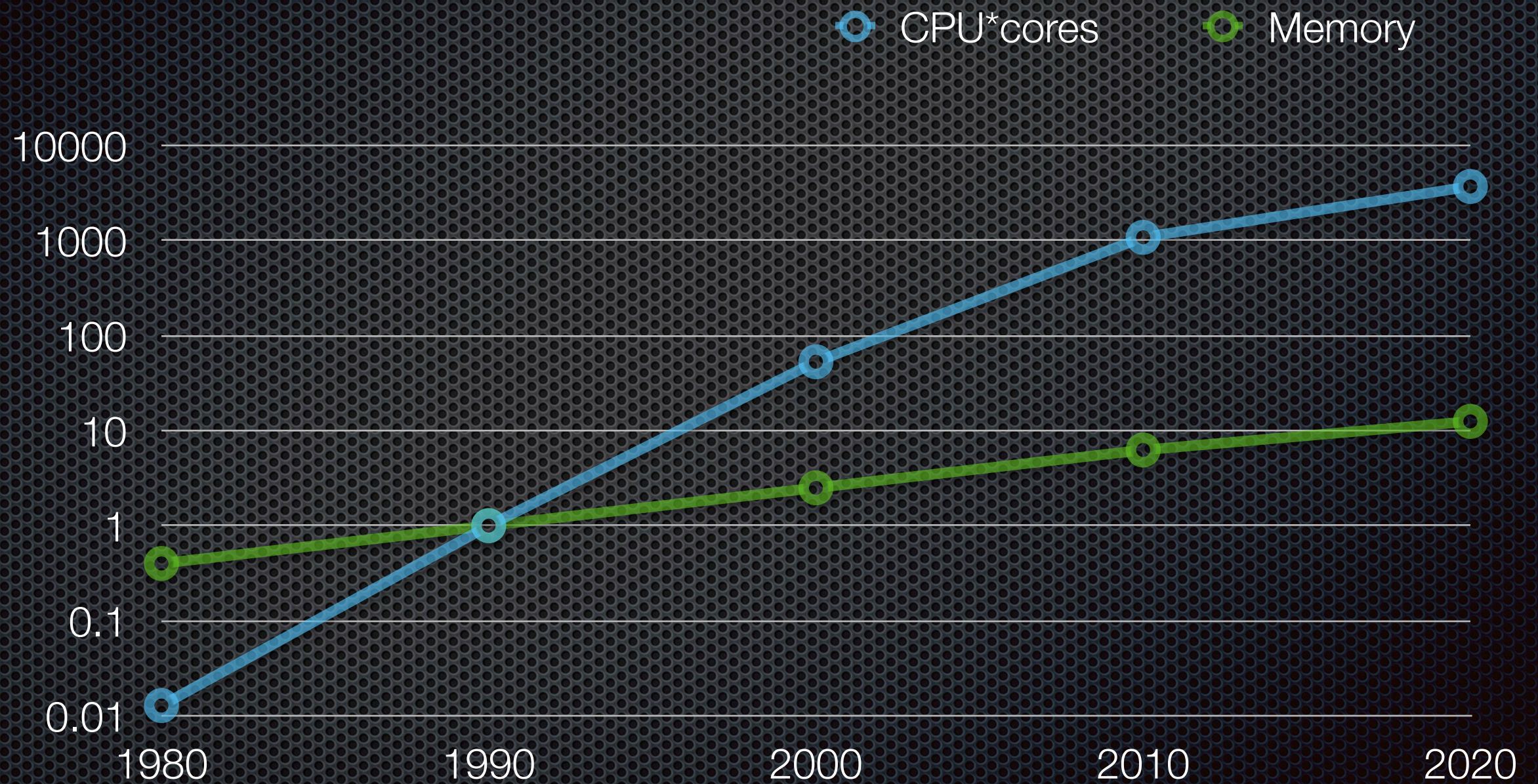
Clock Speed



Clock Speed*core

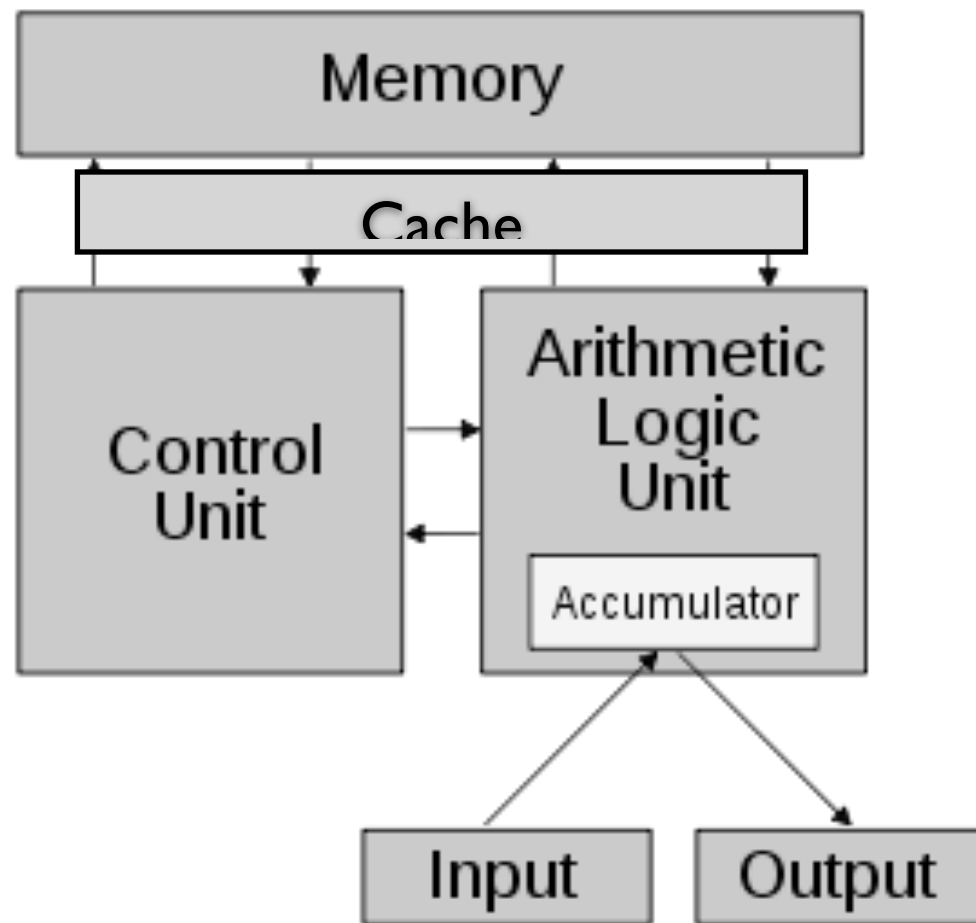


CPU speed vs memory speed



Cache

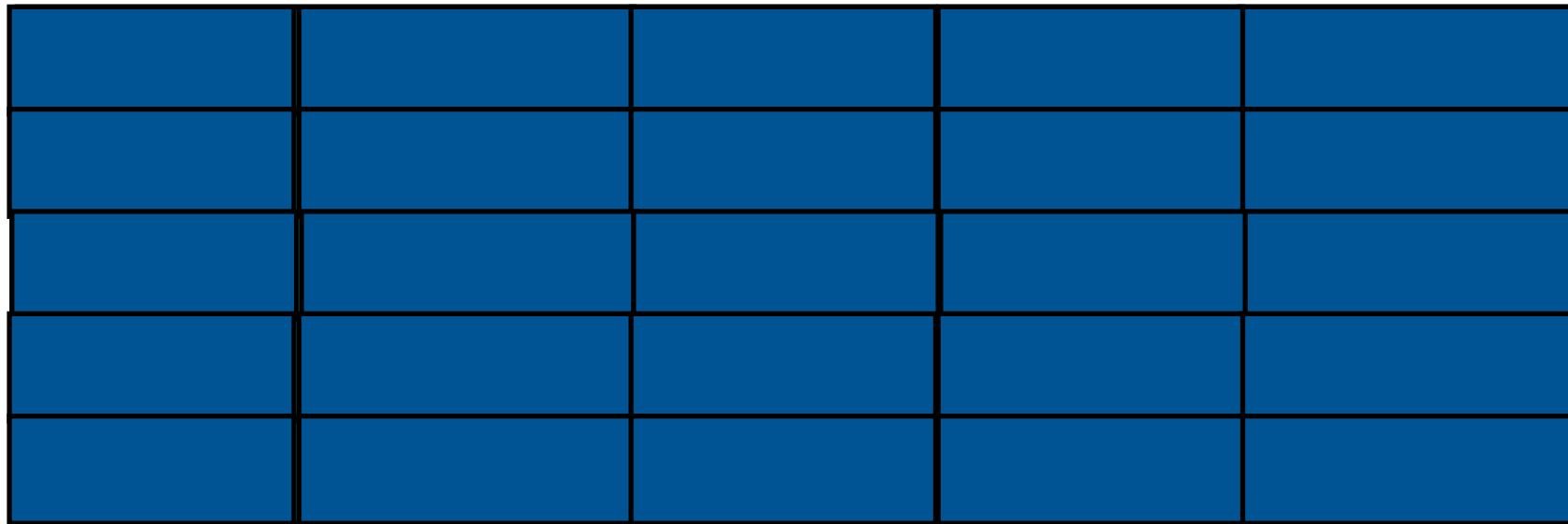
A small memory bank to store data that is accessed frequently. Takes fewer clock cycles to access cache compared to main memory



Wikipedia

Cache basics

Main memory

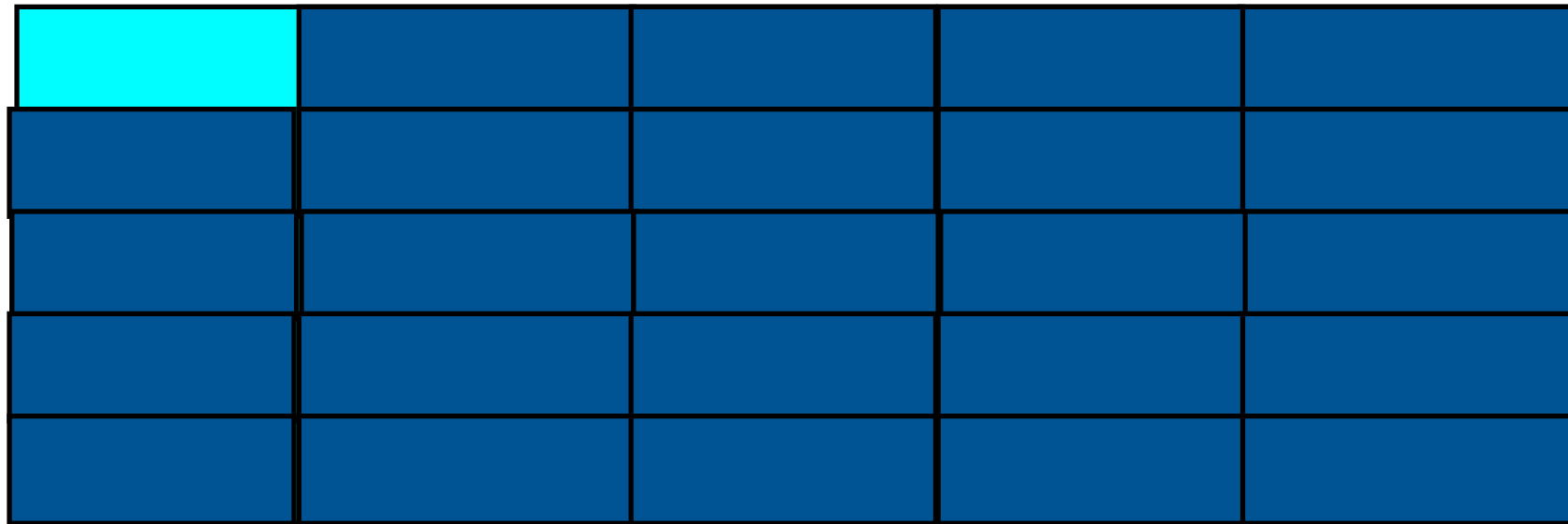


Cache



Cache basics: Request from control unit

Main memory

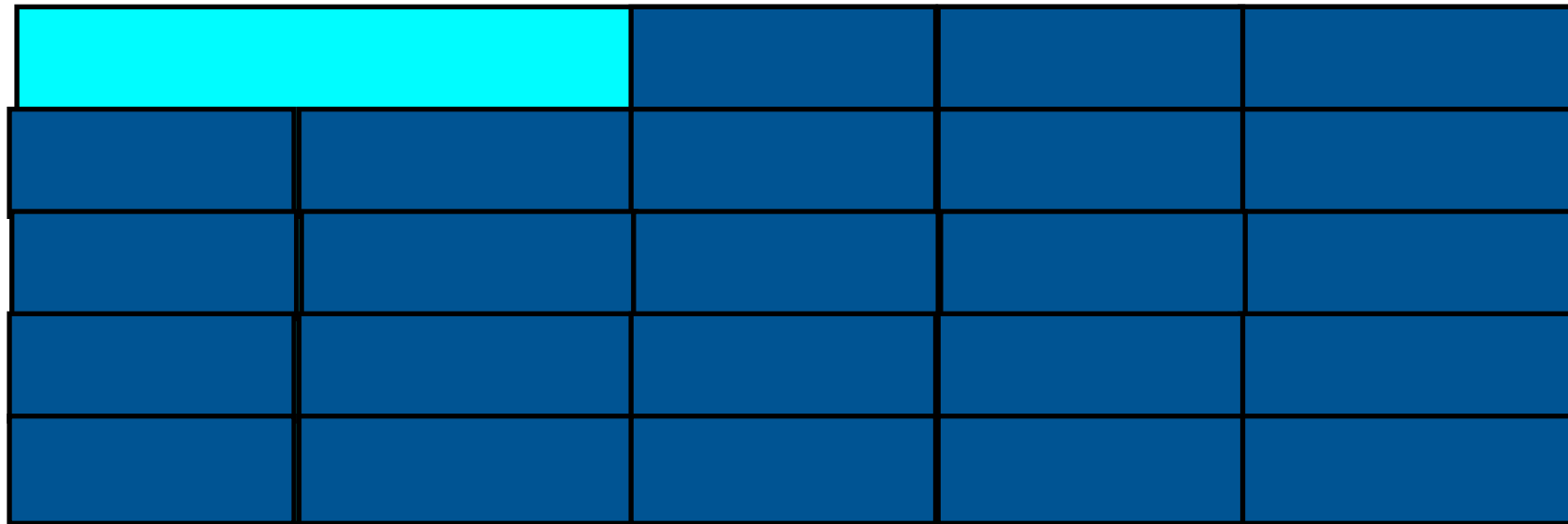


Cache



Cache basics: Request from control unit

Main memory

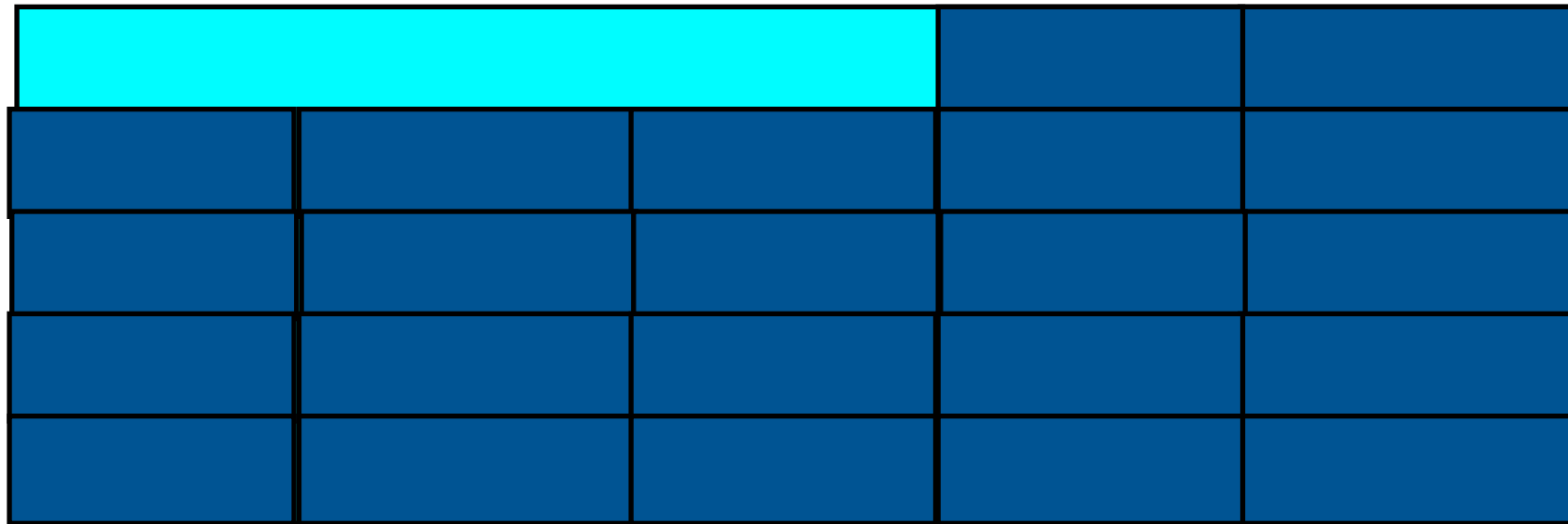


Cache



Cache basics: Request from control unit

Main memory

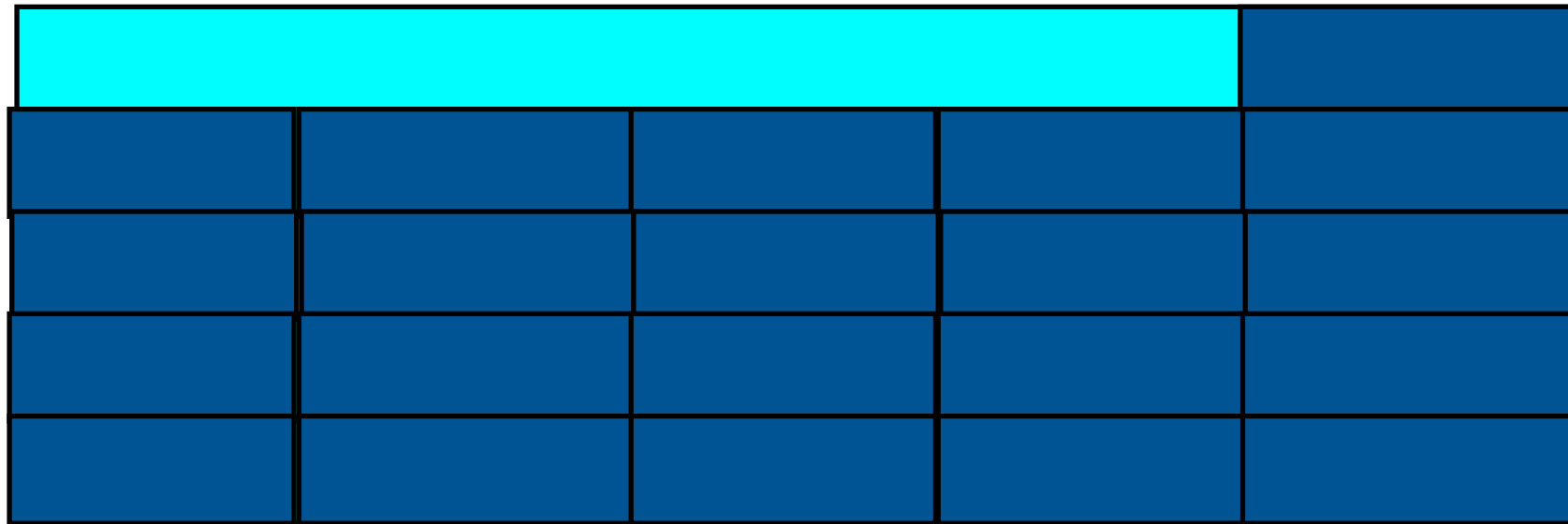


Cache

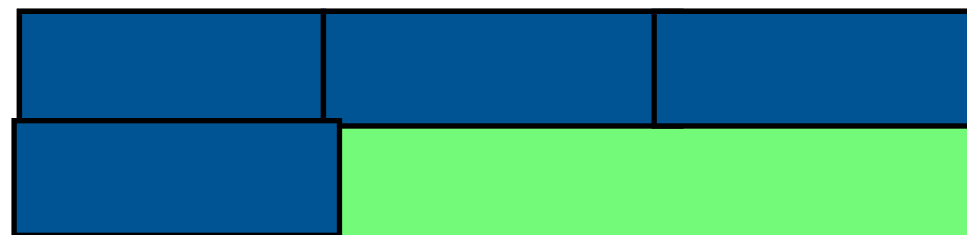


Cache basics: Request from control unit

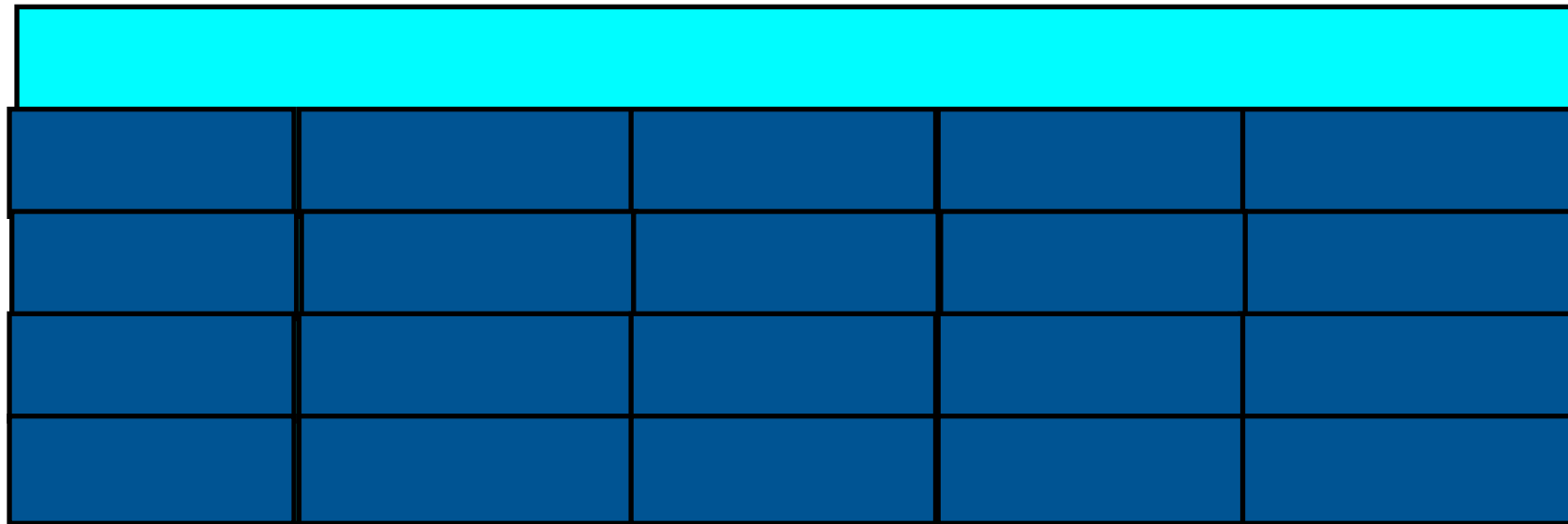
Main memory



Cache



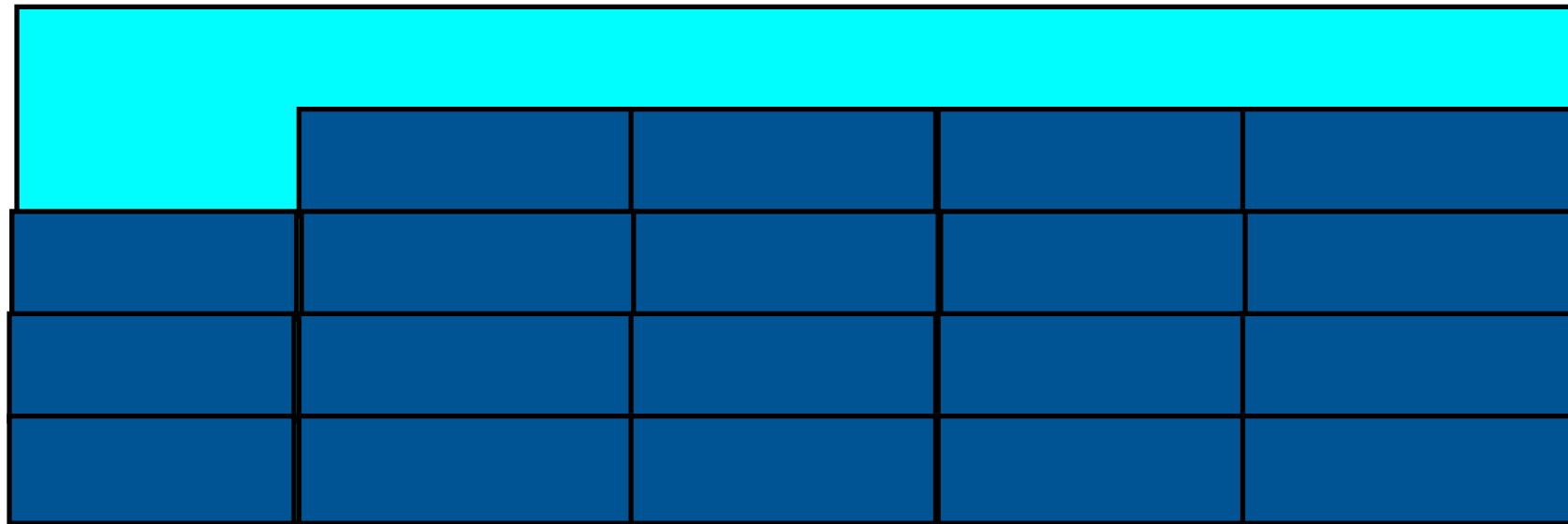
Cache basics: Request from control unit Main memory



Cache



Cache basics: Request from control unit Main memory

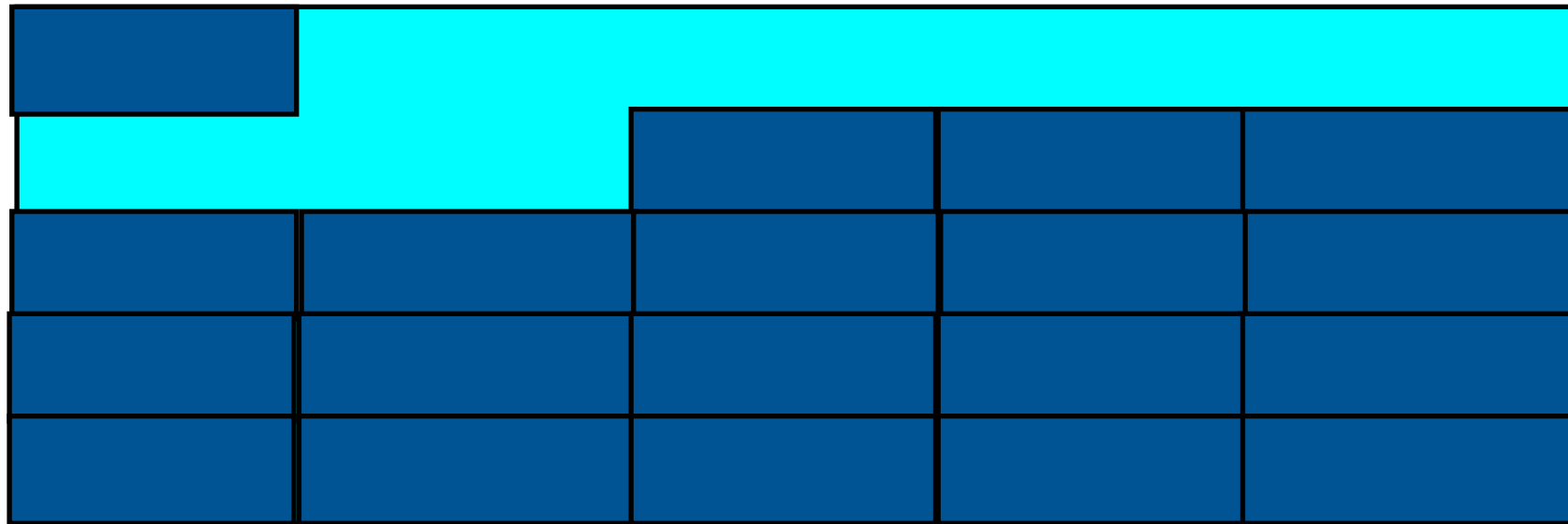


Cache



Cache basics: Request from control unit

Main memory



Cache



Performance problem #1: Memory vs CPU speed growth

- Solution:
 - Add smaller, faster memories often read data

Registers

- Fastest piece of memory
- 1 clock cycle to read/write (~.33-.5 nano seconds)
- Pros
 - Fast
- Cons
 - Small (8-16 registers, 16-32 bit in size)
 - No direct access

L1-Cache

- 32 KB in size
- 3-4 clock cycles (1 ns) to read/write
- Cannot be addressed by the programmer
- Associative (which means it is actually smaller)
- Stored in 64 byte cache line size

L2/L3 Caches

- 256 KB/ 10 MB
- 5 clock cycles (2 ns)/ 10 clock cycles (3ns)
- Cannot be addressed by the programmer
- Again no control by programmer

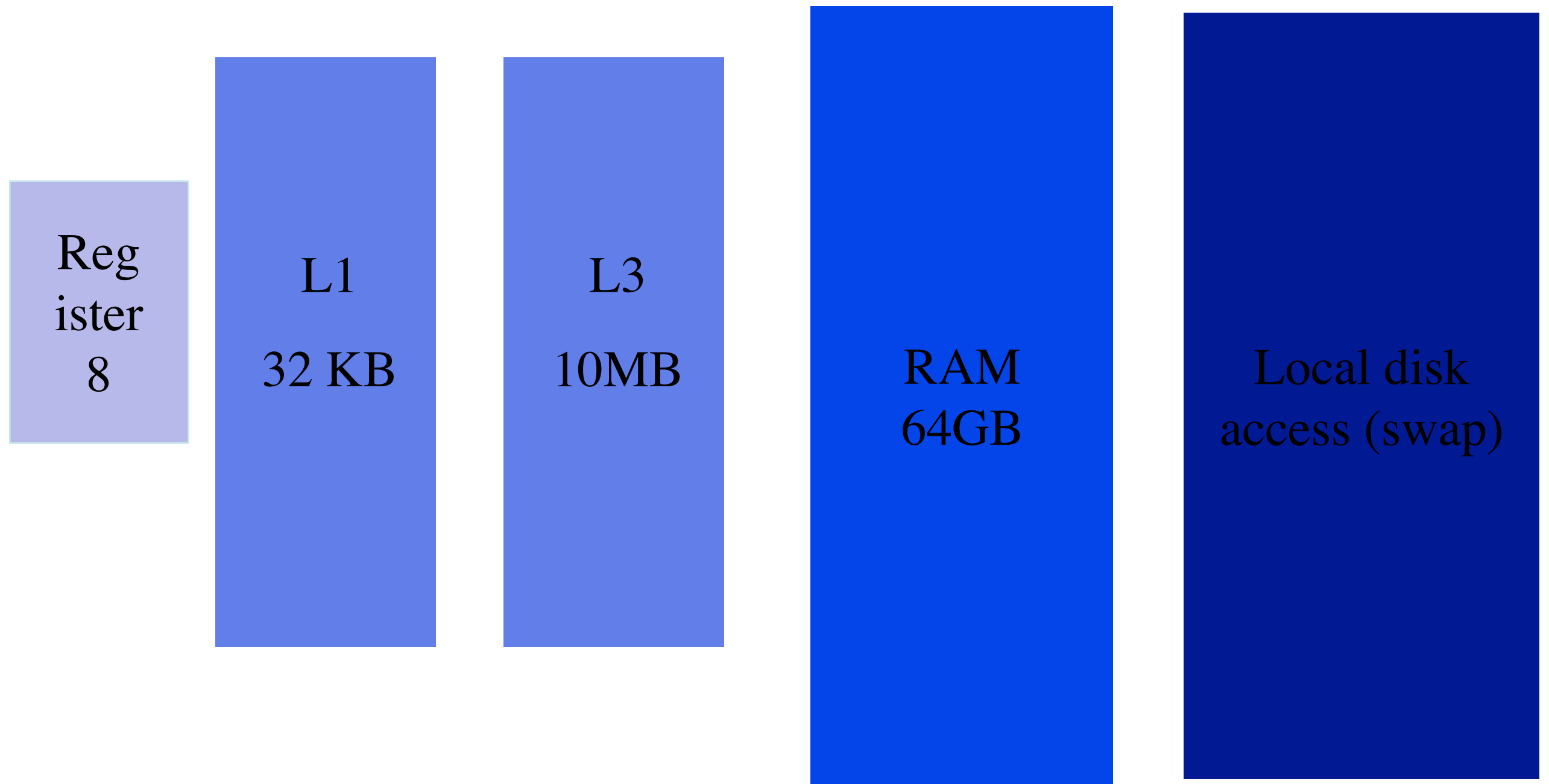
Main memory

- SDRAM (Synchronous dynamic random access memory)
- Large 64 GB
- 60 clock cycles

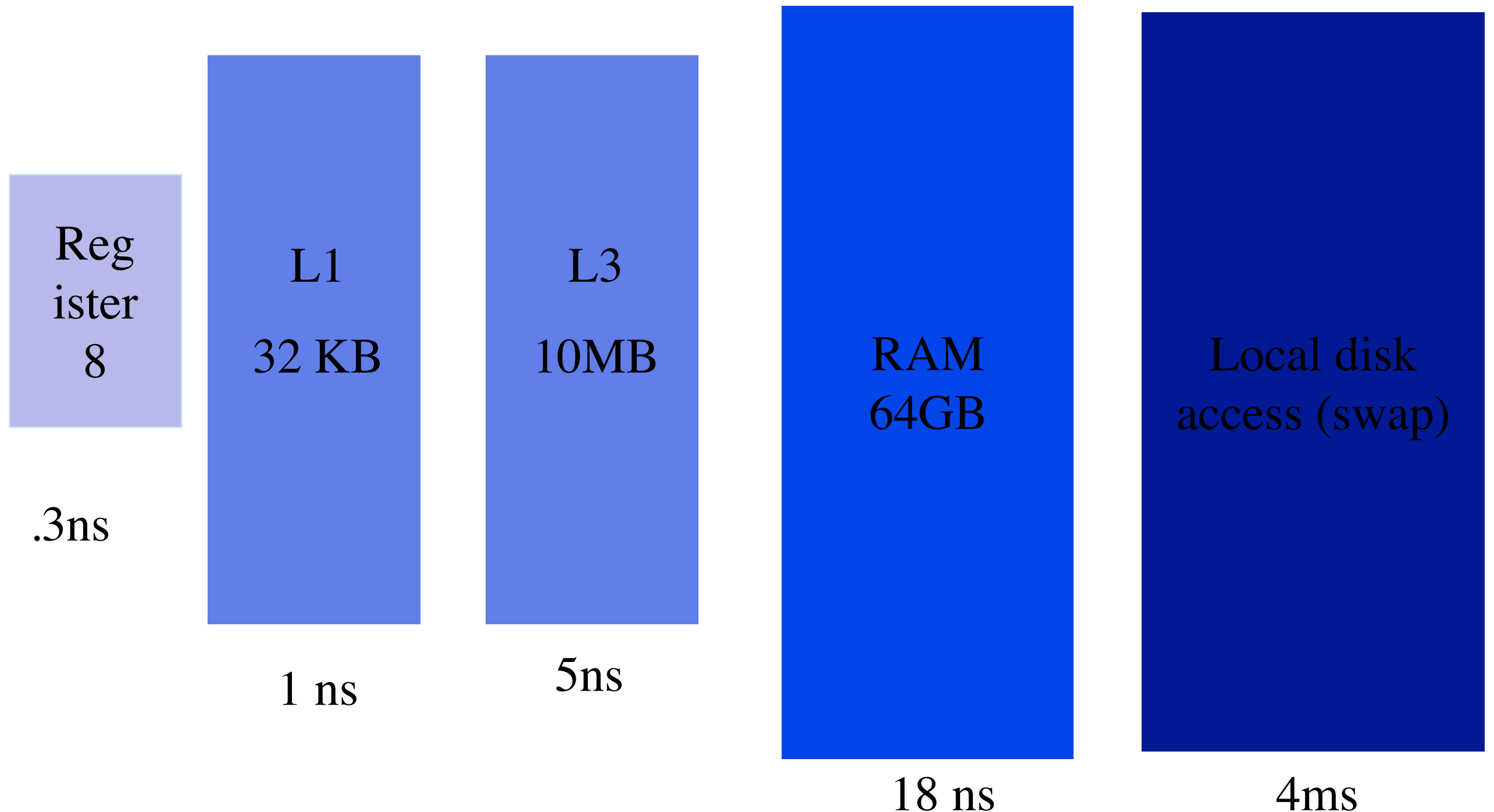
Disk

- Standard disks 7200 rpms
- 120 revolutions per seconds
- 8 ms per revolution
- 4ms read/write from a sector

Further memory options



Further memory options



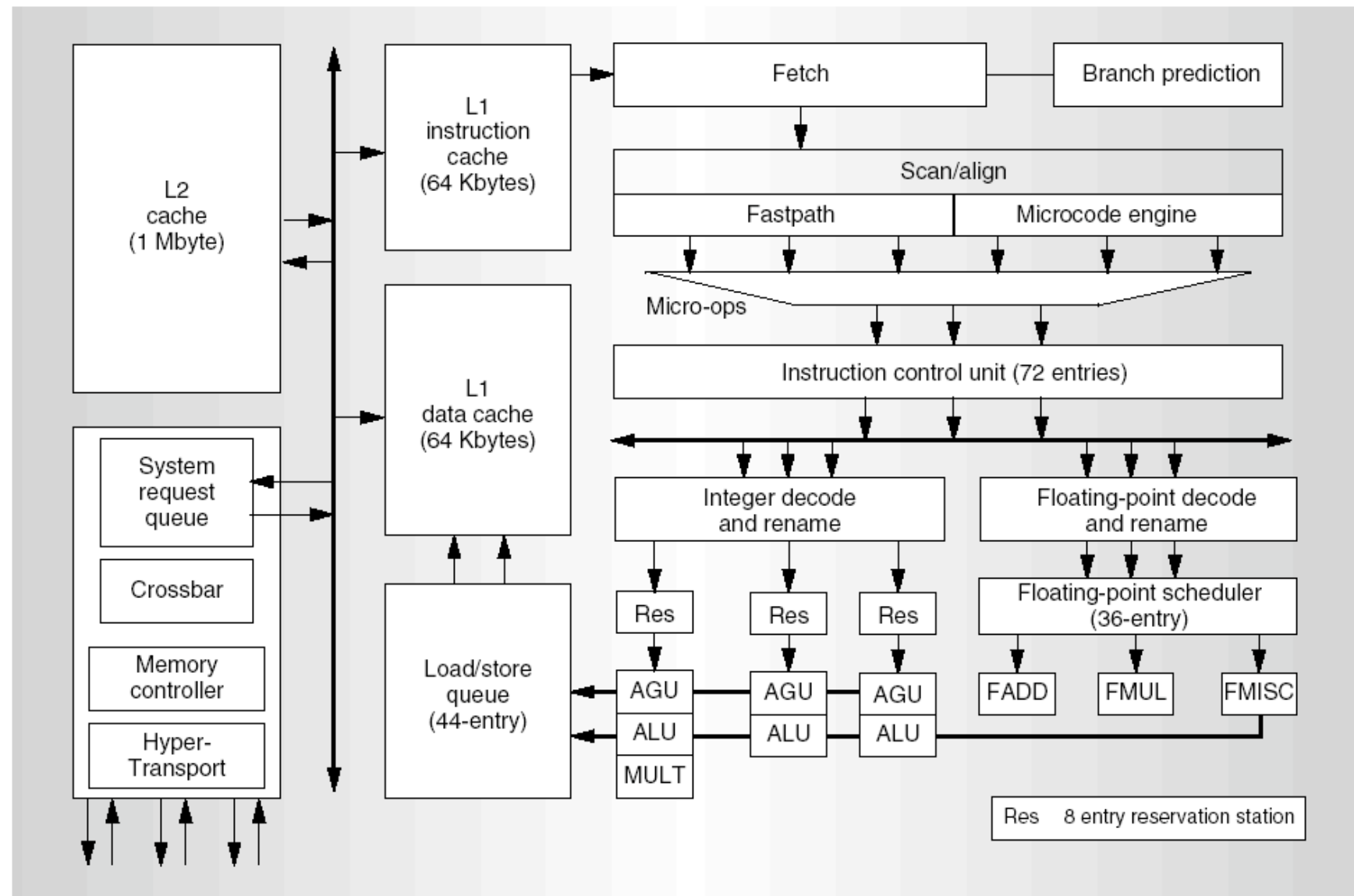
Performance problem #1: Memory vs CPU speed growth

- Solution:
 - Add smaller, faster memories for often read data
 - Copy cache lines instead of single bytes
 - Minimize latency problems
 - Allow parallel memory banks

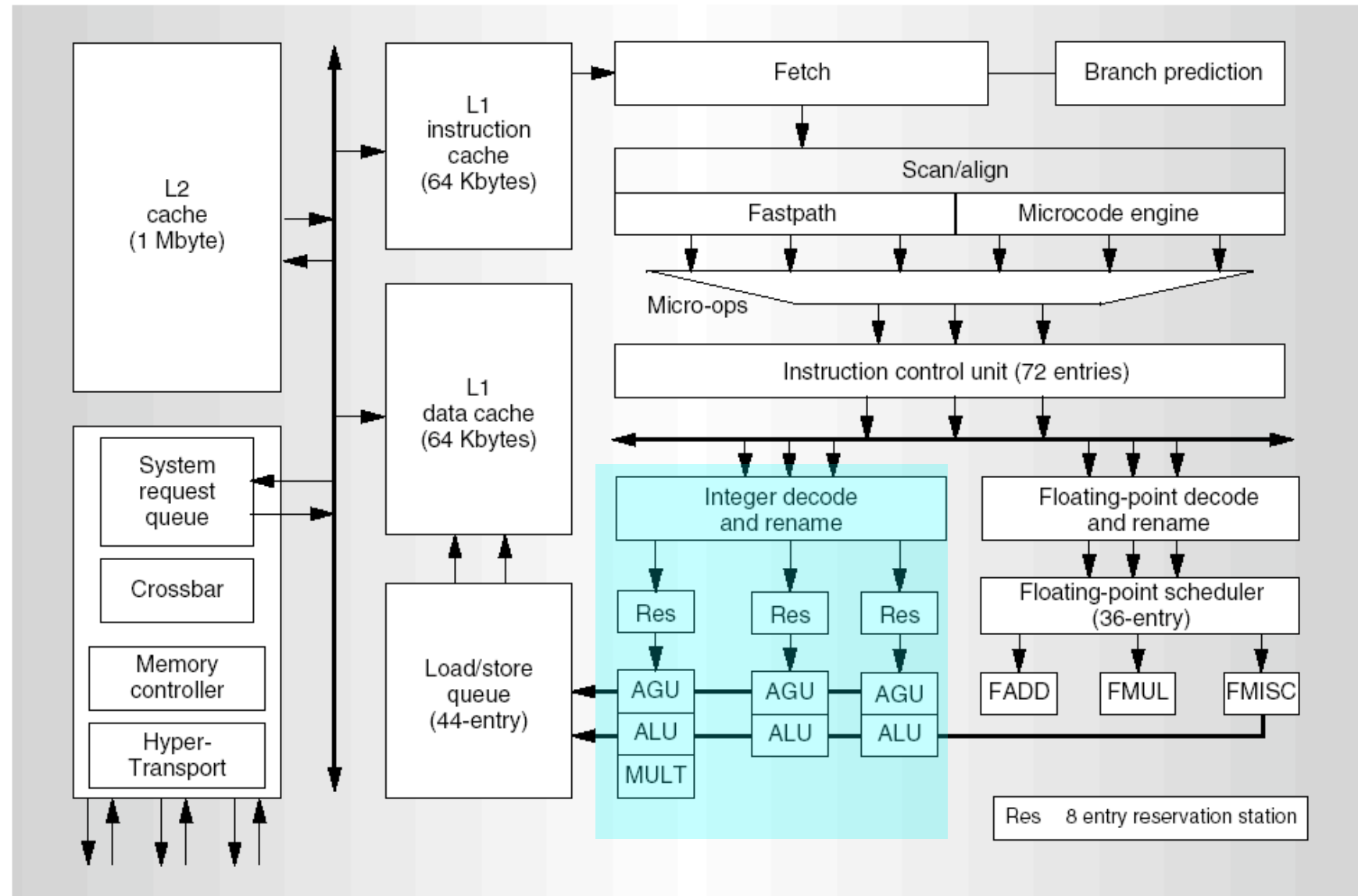
Locality: Performance maximization

- Spatial locality:
 - Use a high percentage of each cache line read from memory
- Temporal locality
 - Reuse cache line before it leaves caches

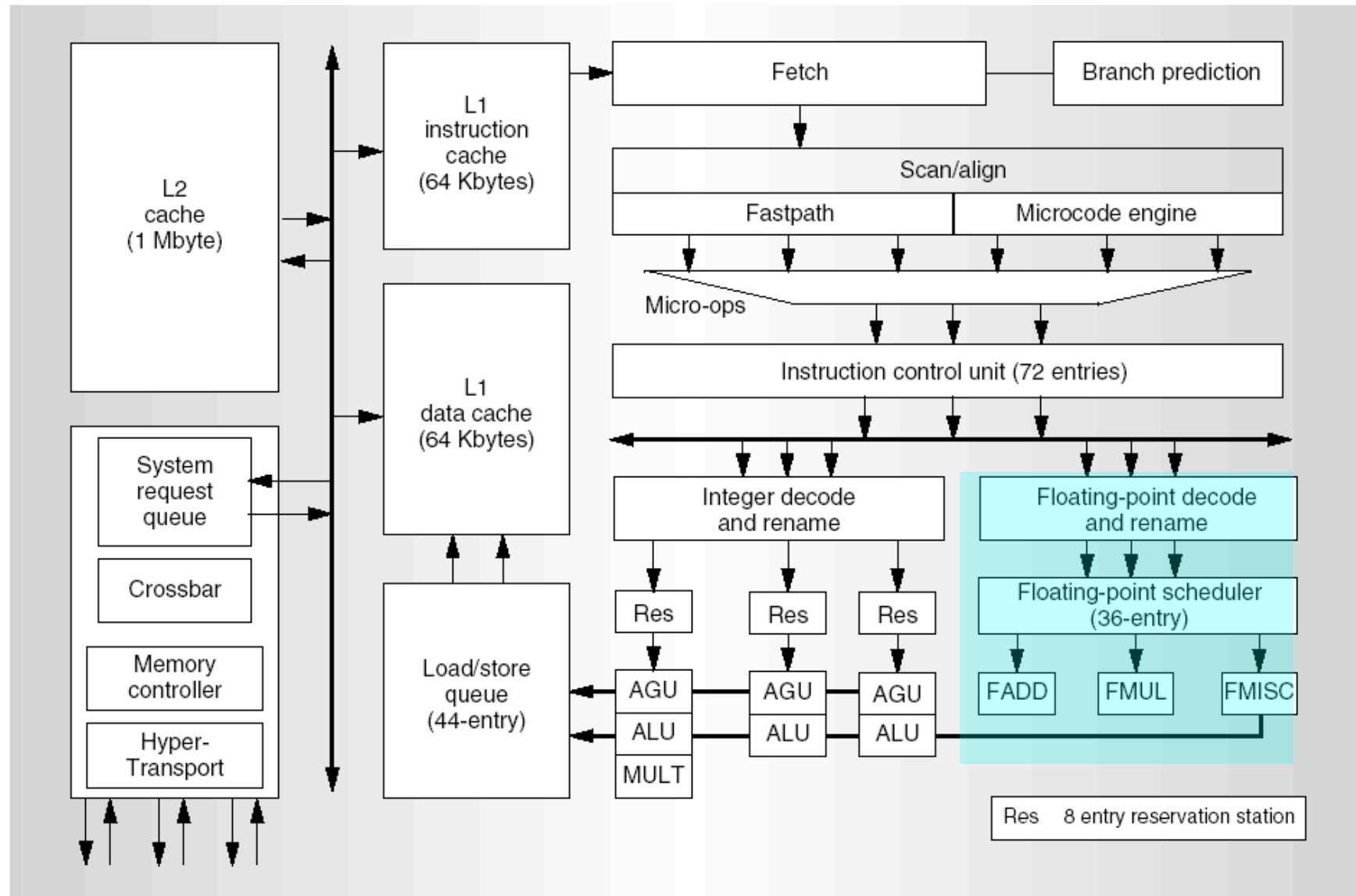
Opteron processor



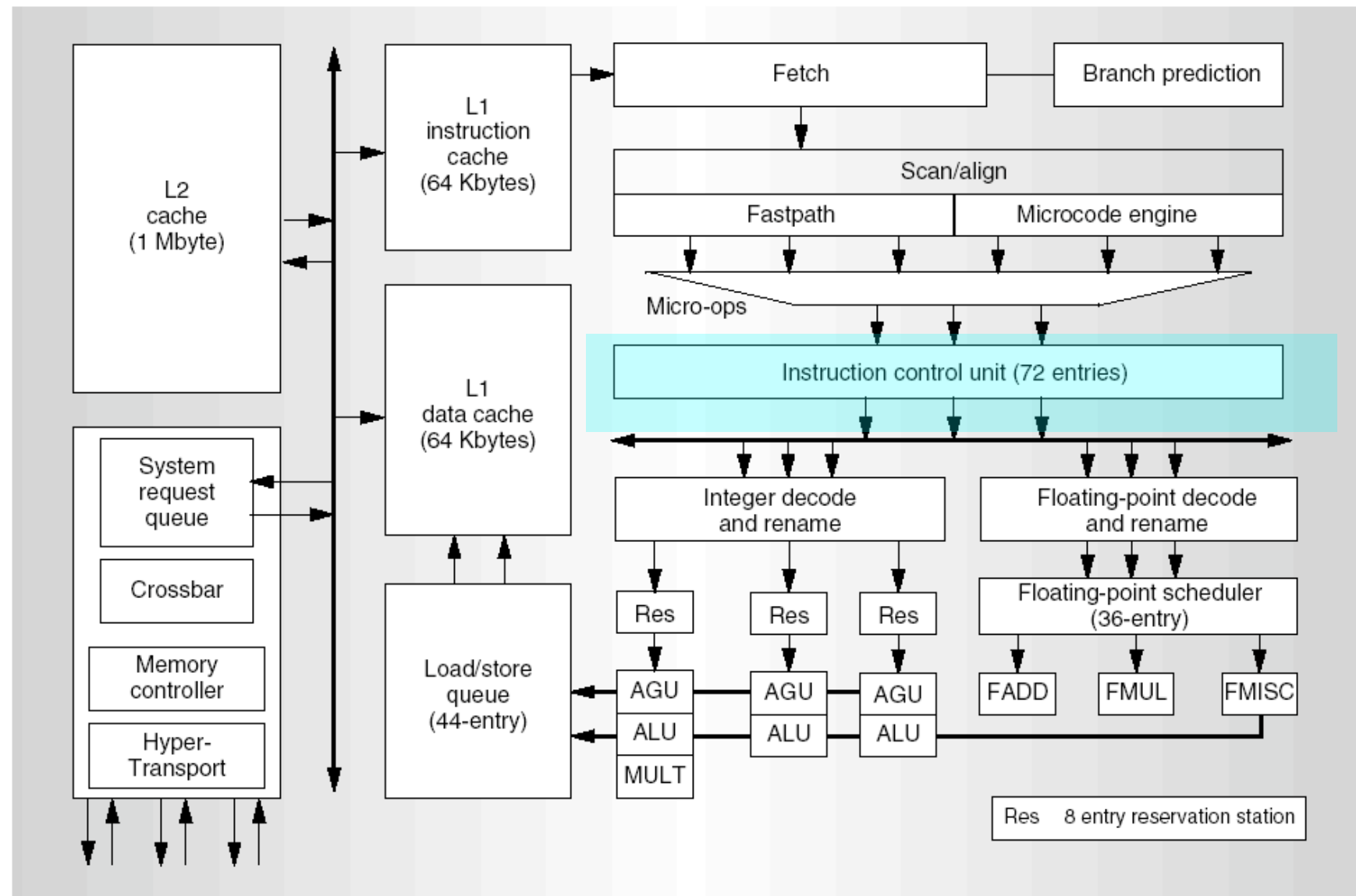
Integer operations



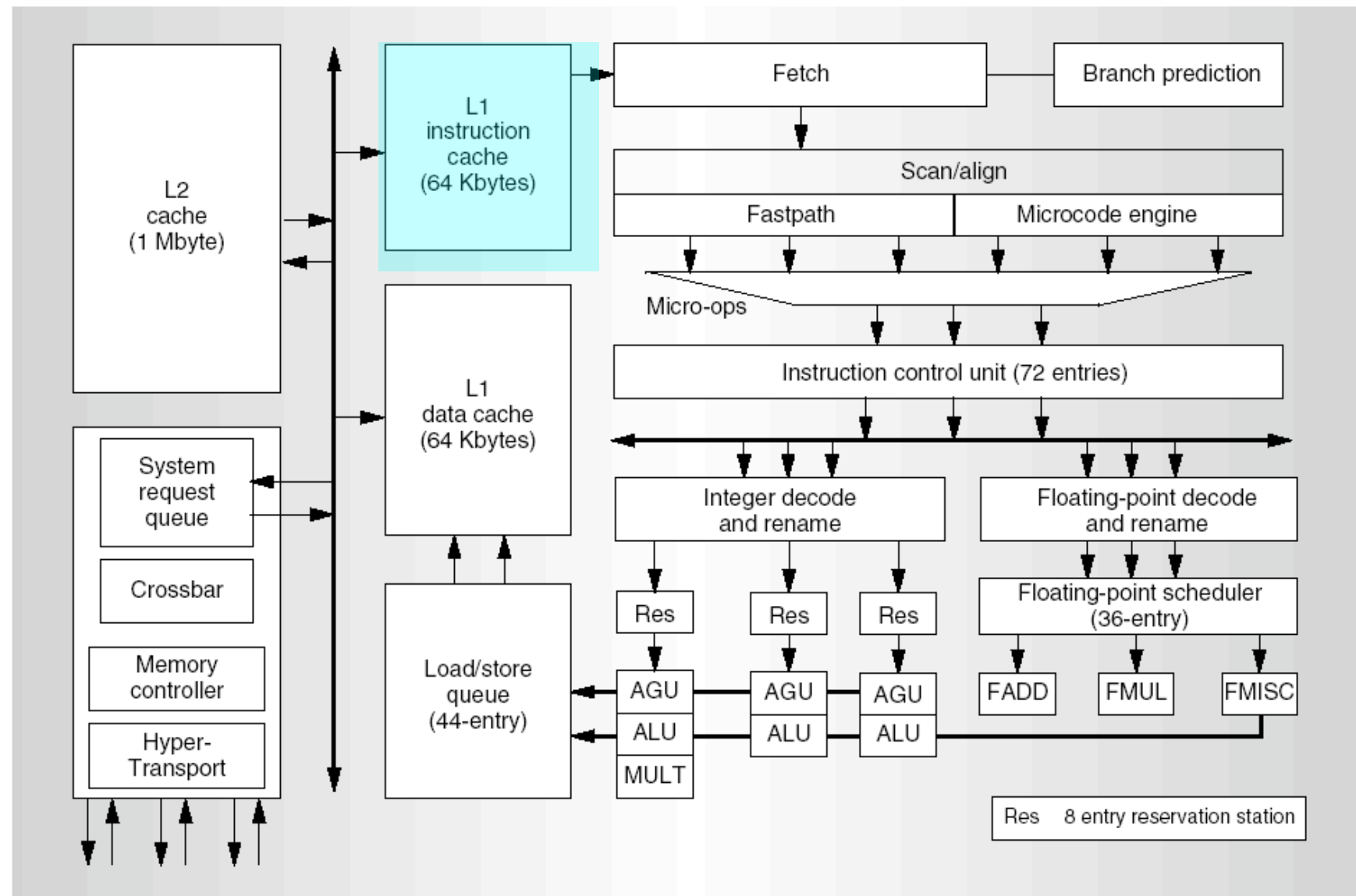
Float operations



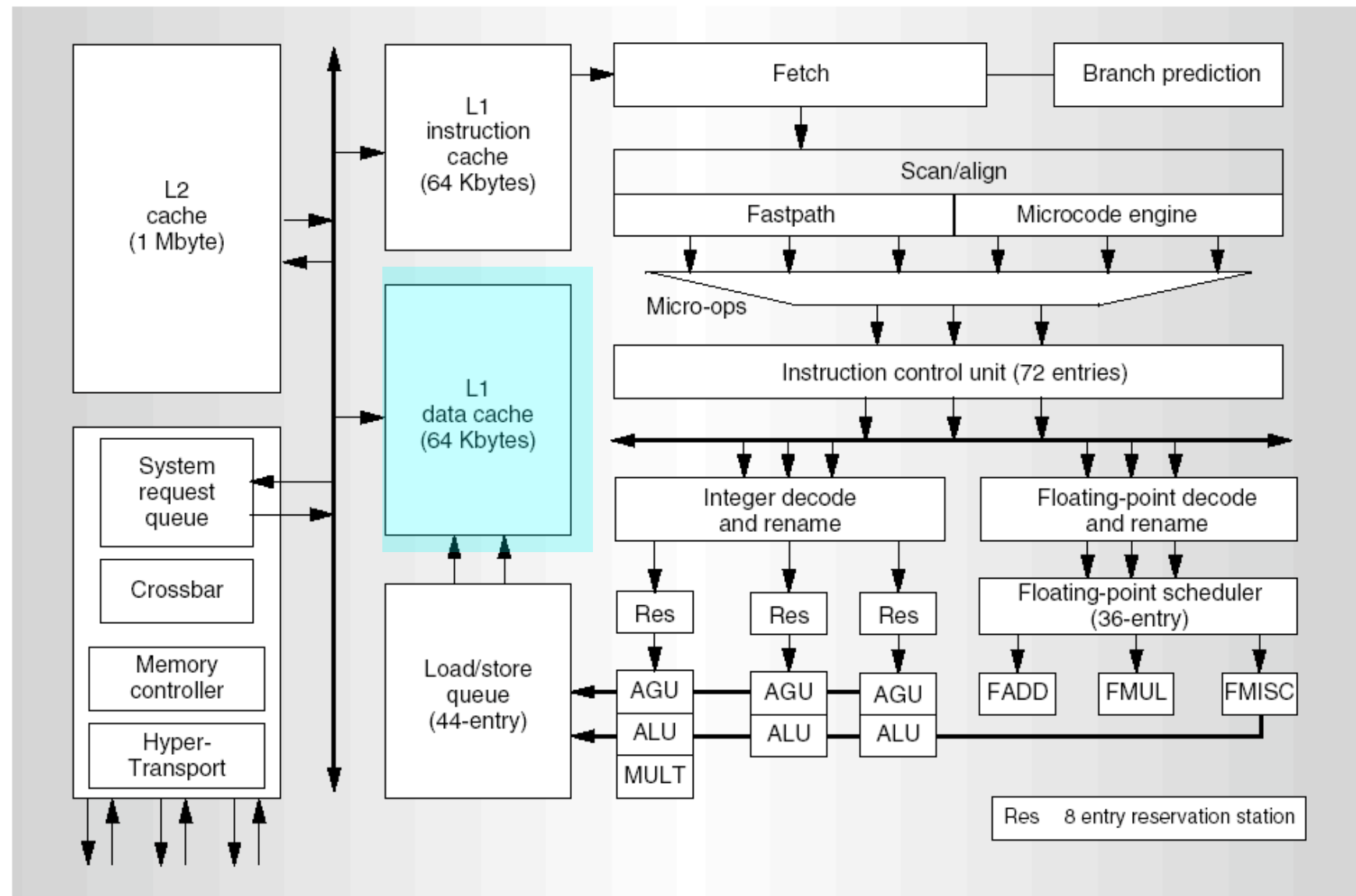
Instruction handling



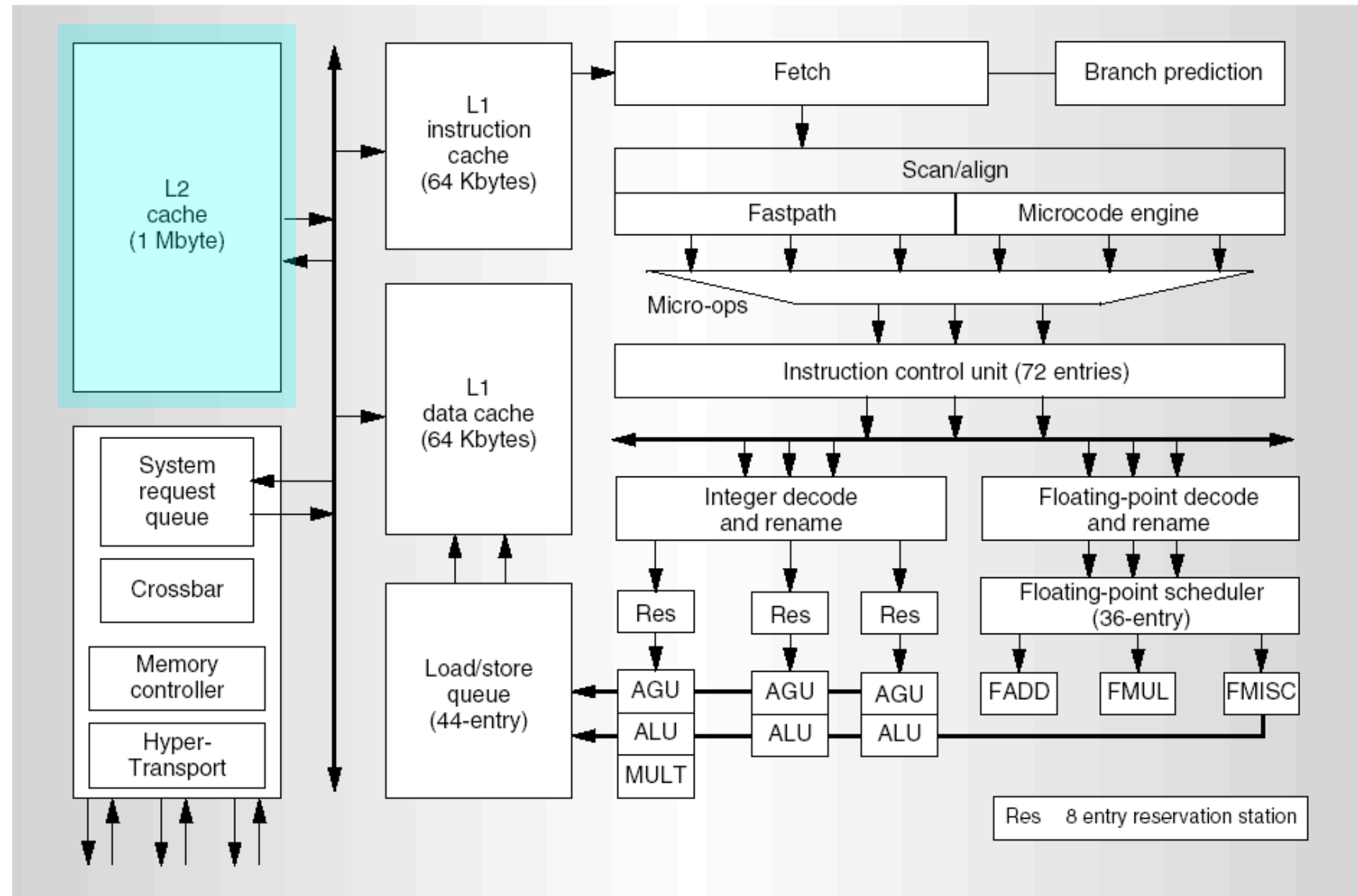
Fast access to instructions



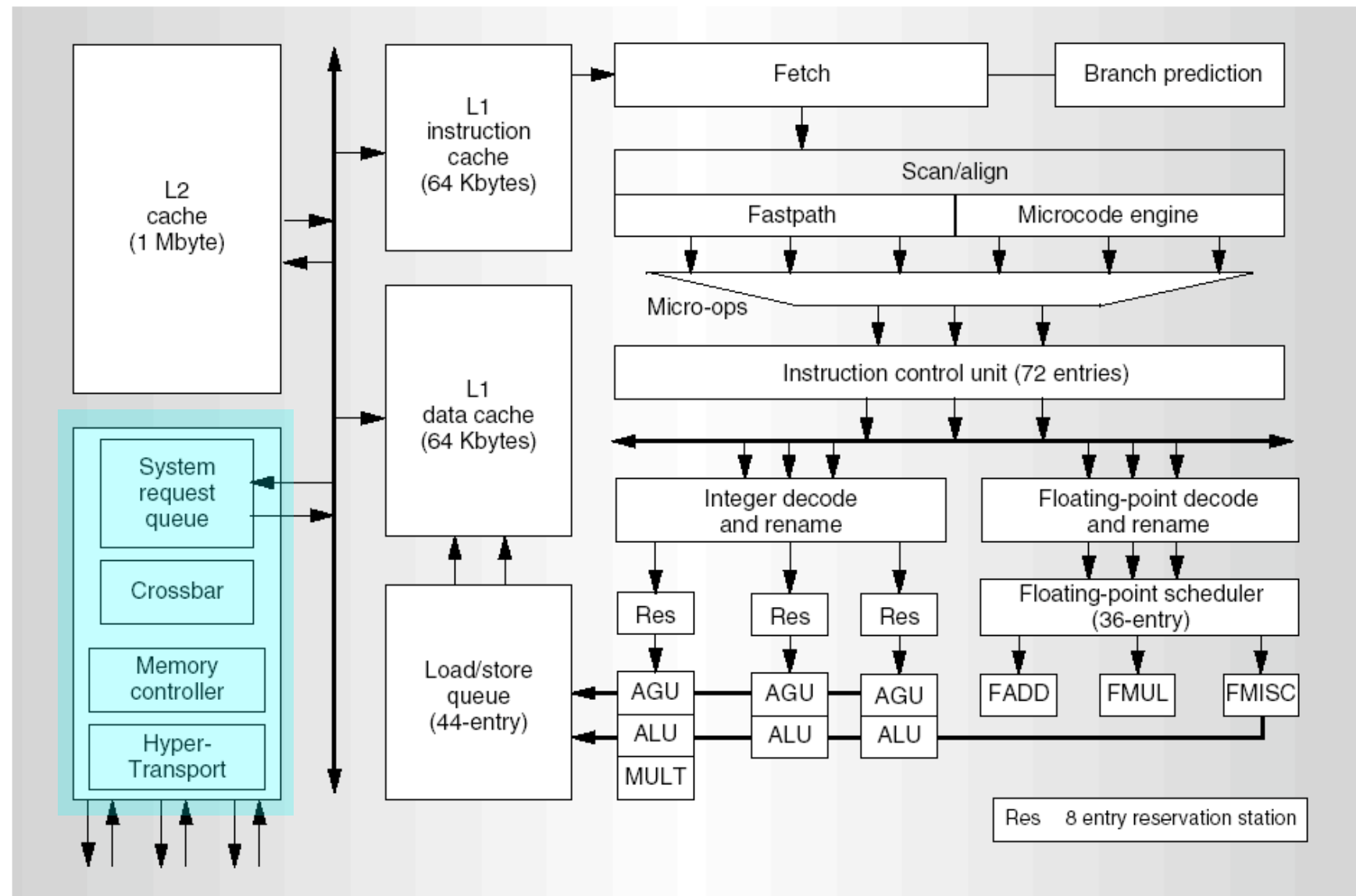
Fastest access to memory



Fast access to memory



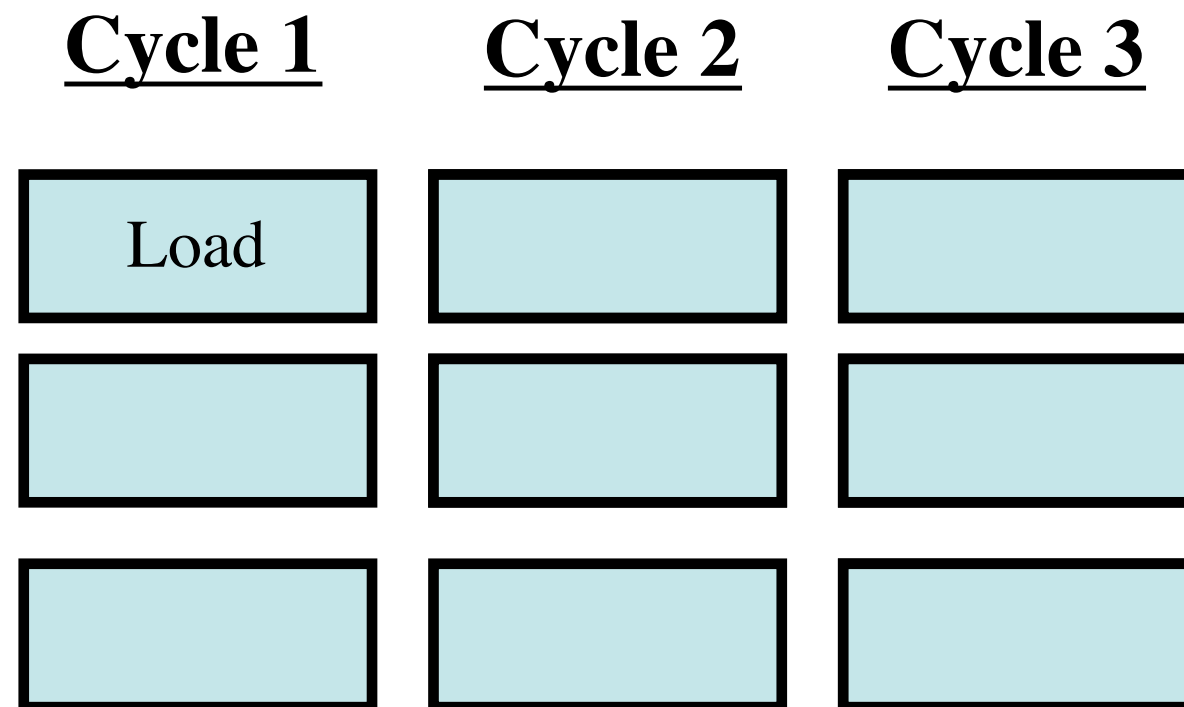
Access to RAM



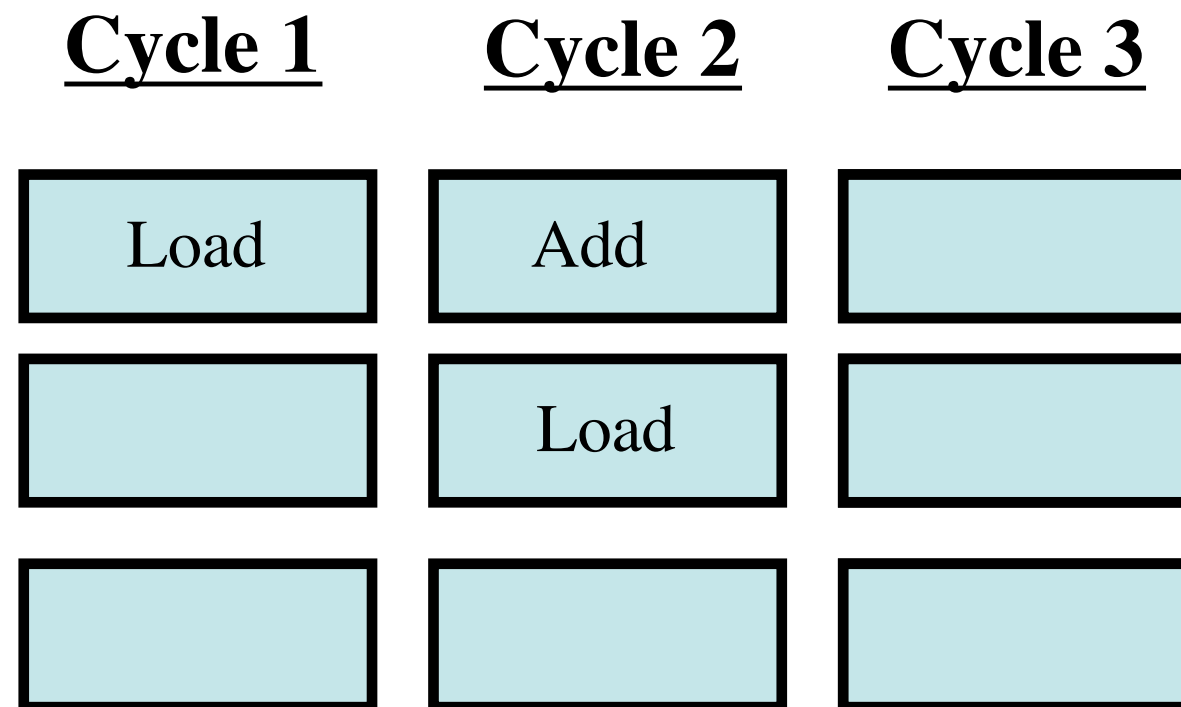
Instruction pipelining

- Adding two numbers uses different portions of the CPU at each clock step
- We can gain a significant speedup by performing the operations in parallel rather than sequentially

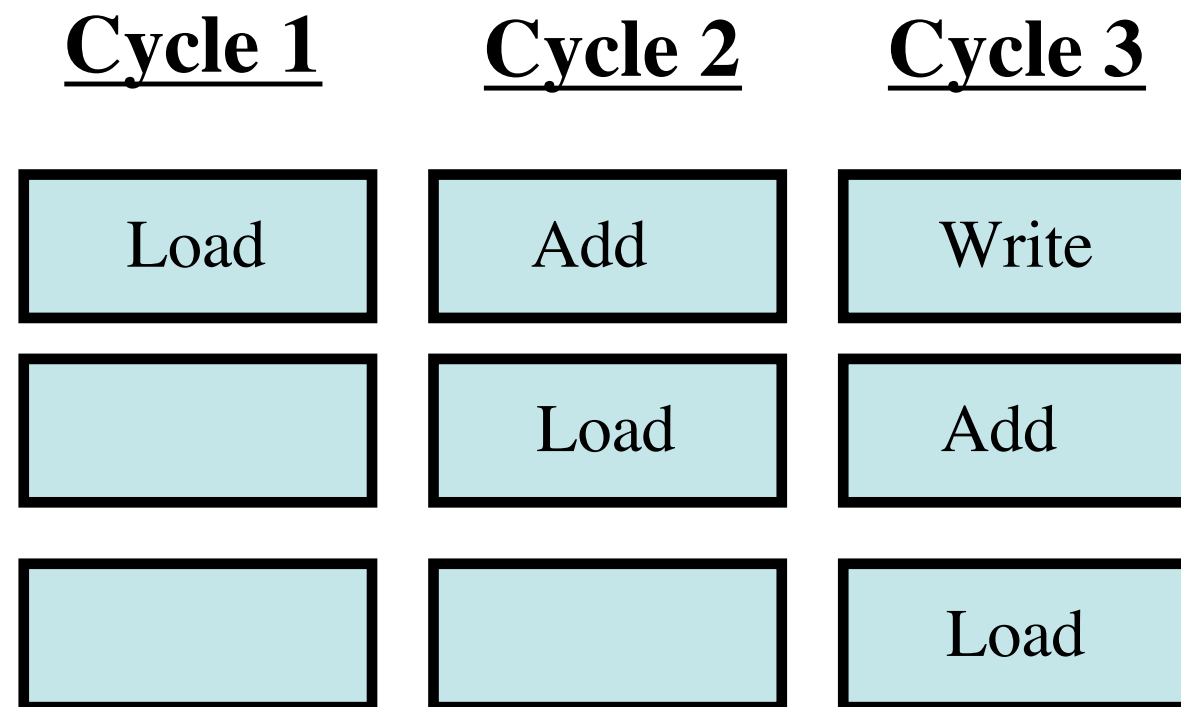
Adding two numbers using instruction pipelining




Adding two numbers using instruction pipelining



Adding two numbers using instruction pipelining





```
For i=0:n  
For j=0:n  
    print(i,j)  
    out[i,j]=in[i,j]^2
```