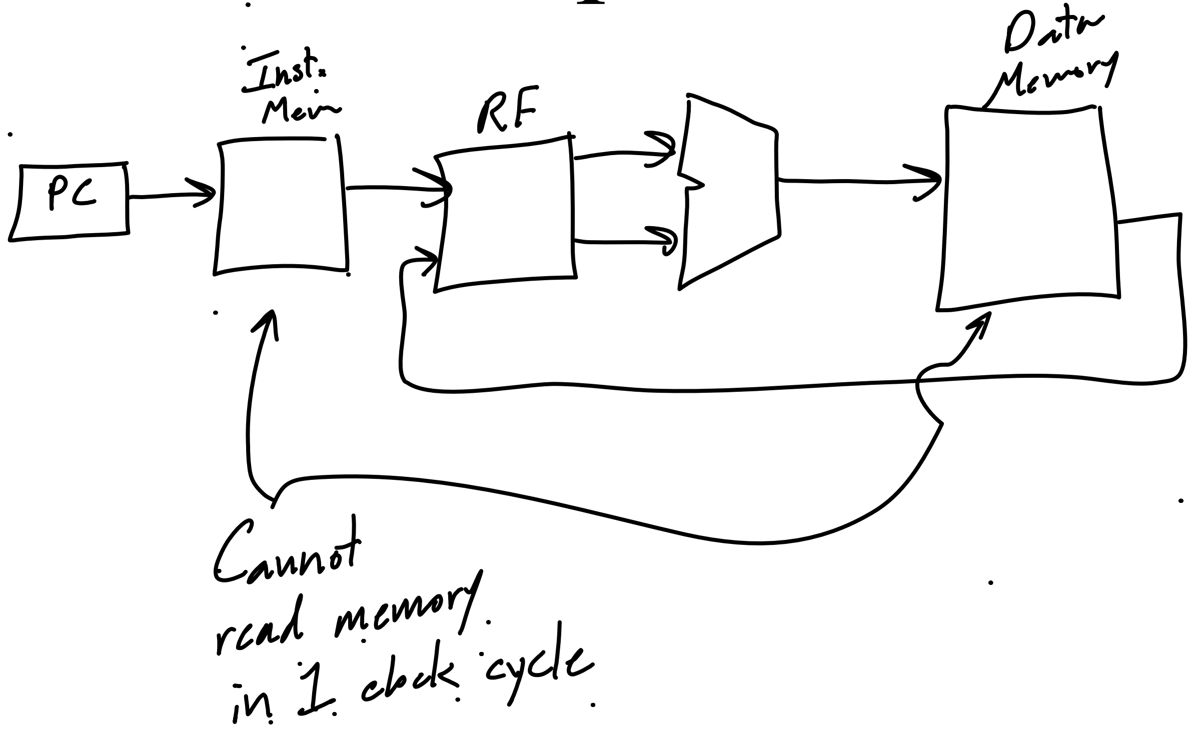
Caching

Ch. 5.

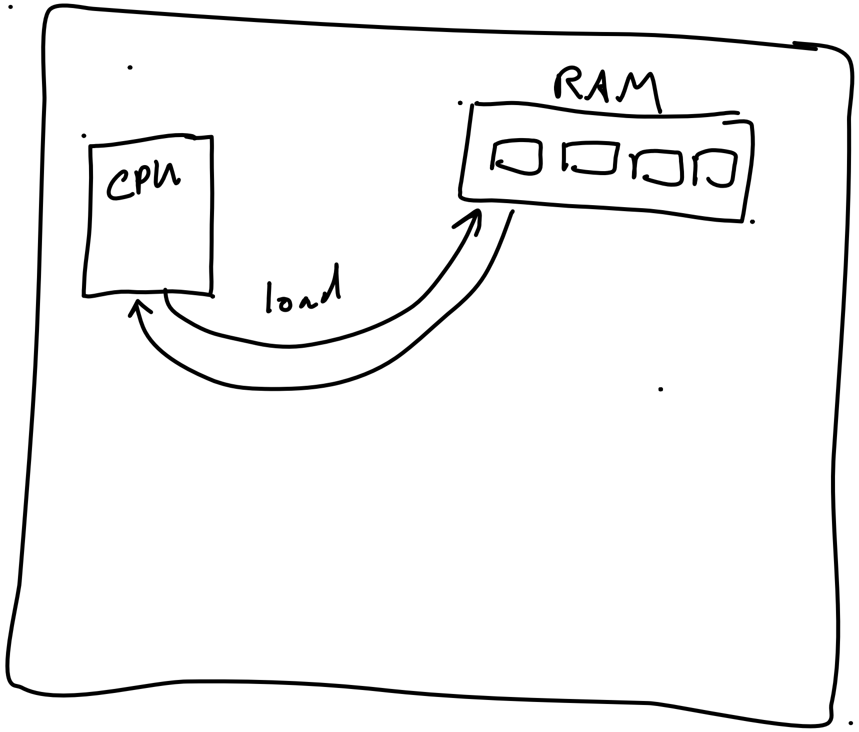
Single-cycle and pipelined datapaths



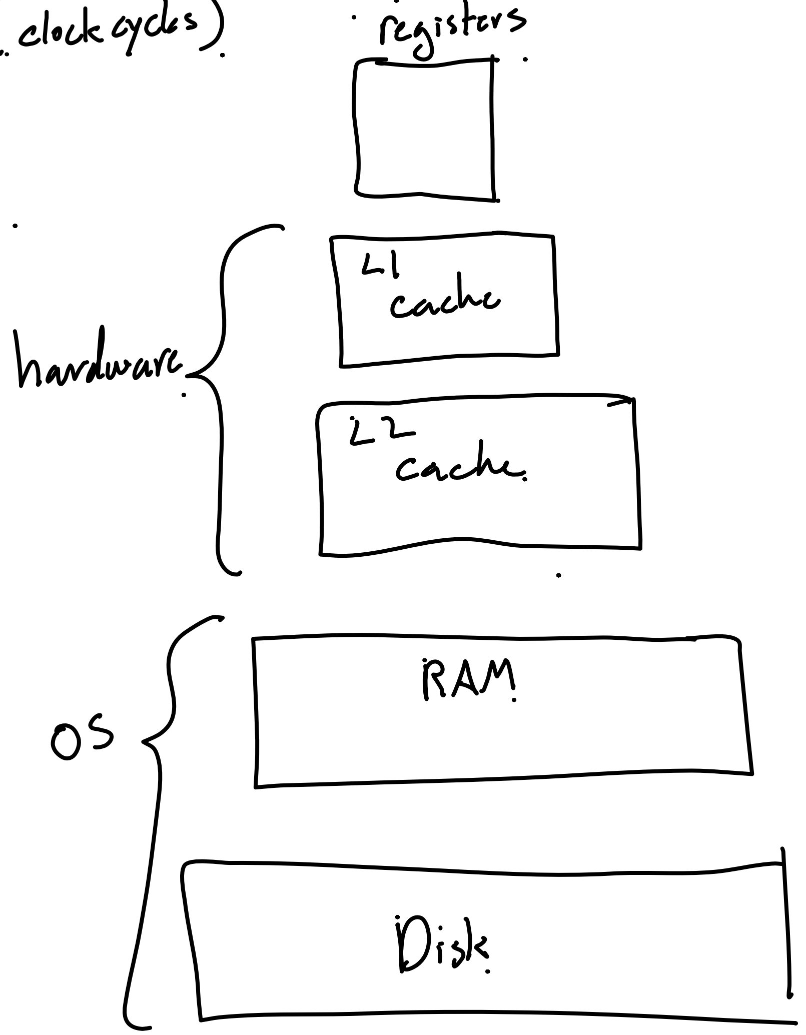
CPU speed vs. Memory speed

$\downarrow$ 2 GHz - 3 GHz

$\downarrow$ 1066 MHz

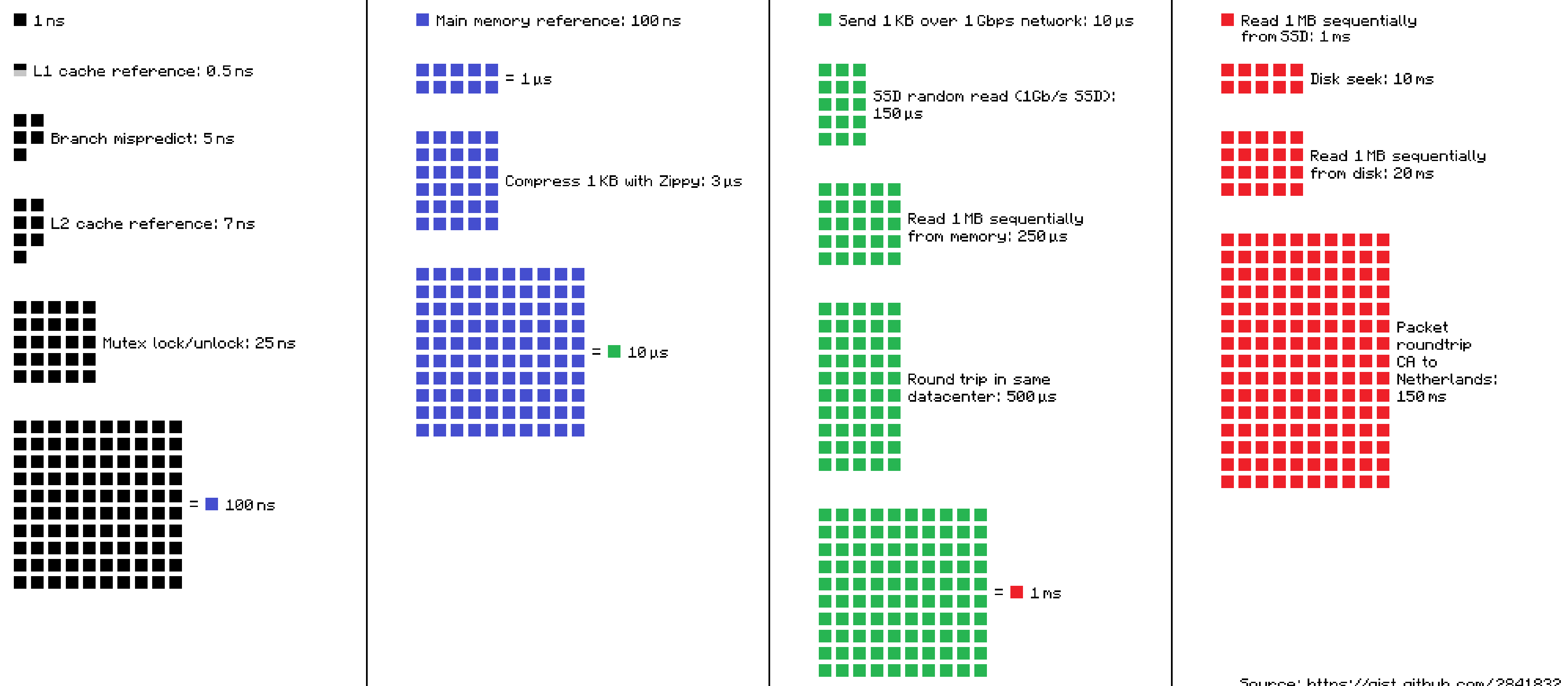


Memory Hierarchy



Speed (CPU clock cycles)		Size	Cost/bit
1	registers	32 registers x 32 bits	expensive
4	L1 cache	32KB- 128KB	less expensive
10-12	L2 cache	2MB- 8MB	even cheaper
40-200	RAM	2GB- 8GB	cheap
SSD: 150,000 hard drive: 3M	Disk	1TB- 5TB	dirt cheap

Latency Numbers Every Programmer Should Know



Source: https://gist.github.com/2841832

Locality

$\hookrightarrow$ Programs tend to use the same data/inst. that were used recently

```
loop: ... X
      ... X
      ... X
      ... X
  beq ..., ..., loop
```

Program Characteristic

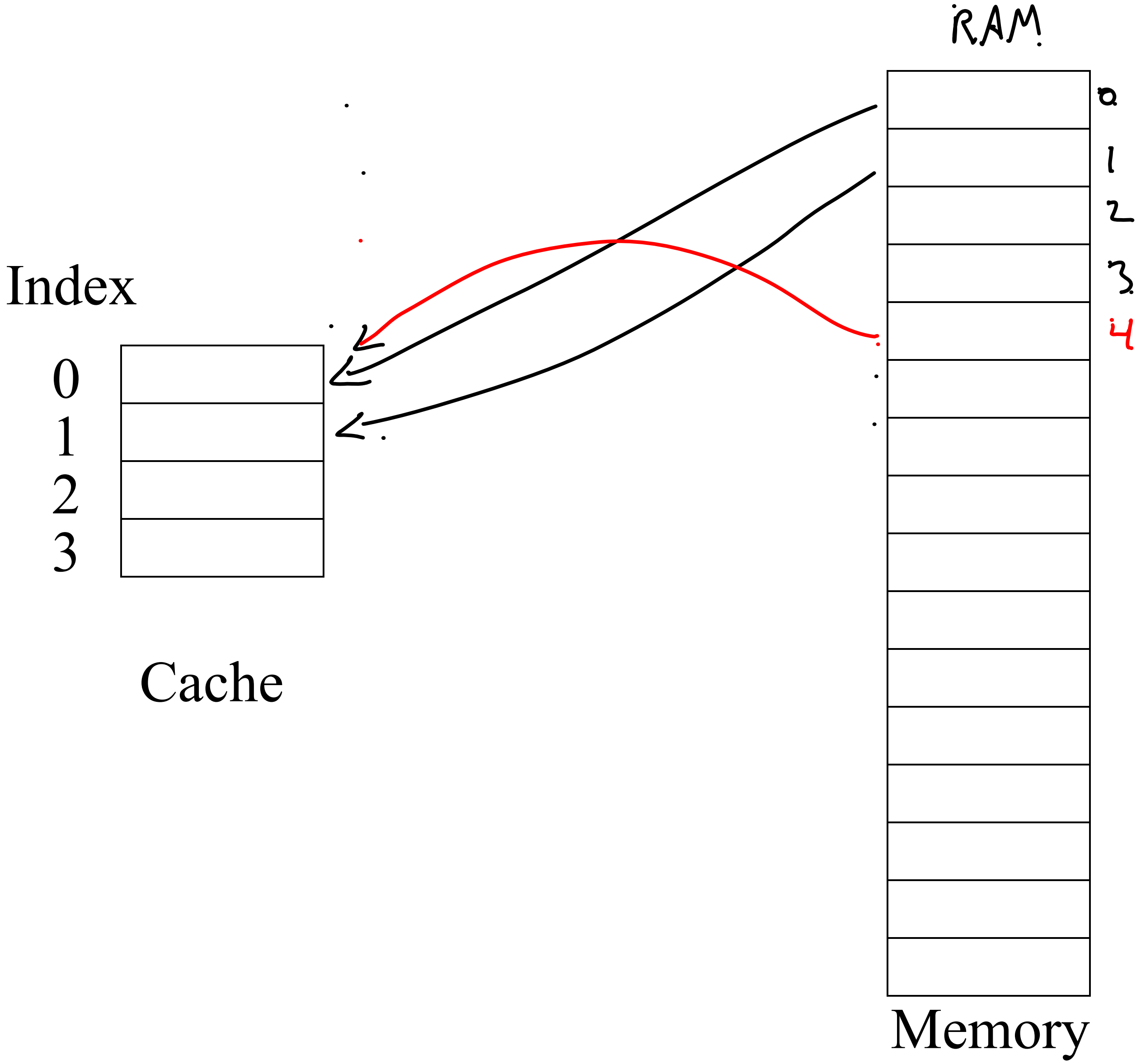
- Temporal Locality – items accessed recently are likely to be accessed in the near future
  (Time)
- Spatial Locality – if an item is accessed, a program will likely access its neighbor
  (Space)

Cache Design to use locality

- Temporal Locality

- Spatial Locality

Placing data in a cache



Definitions

- Block (Line) – the unit of data transferred into a cache
- Hit – data is found in the cache
- Miss – data is not found

- Hit time (Access time) – how long to get the data

- Miss Penalty – time to receive the data from a lower level

Accessing a cache

Index = Address % 4

Address stream:

Addr:	1	4	7	1	5	1	4	7	8	7
Index:	1	0	3	1	1	1	0	3	0	3
	M	M	M	H	M	M	H	H	M	H

Index	
0	~~4~~ 8
1	~~1~~ ~~5~~ 1
2	
3	7

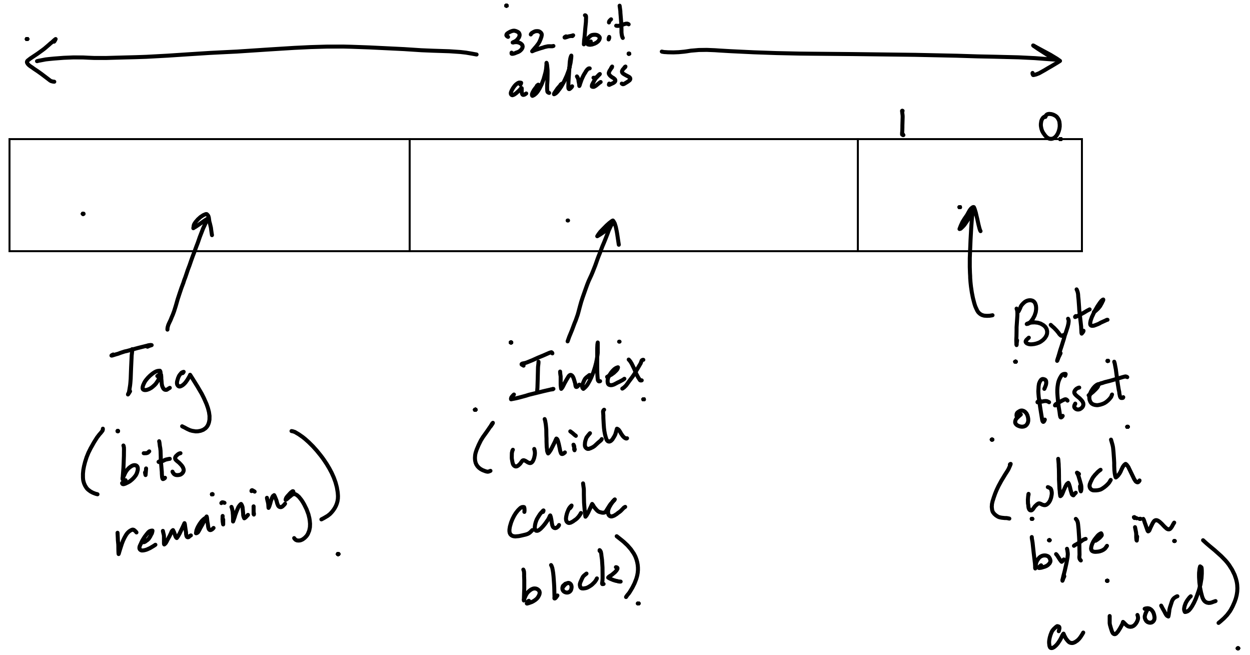
Cache

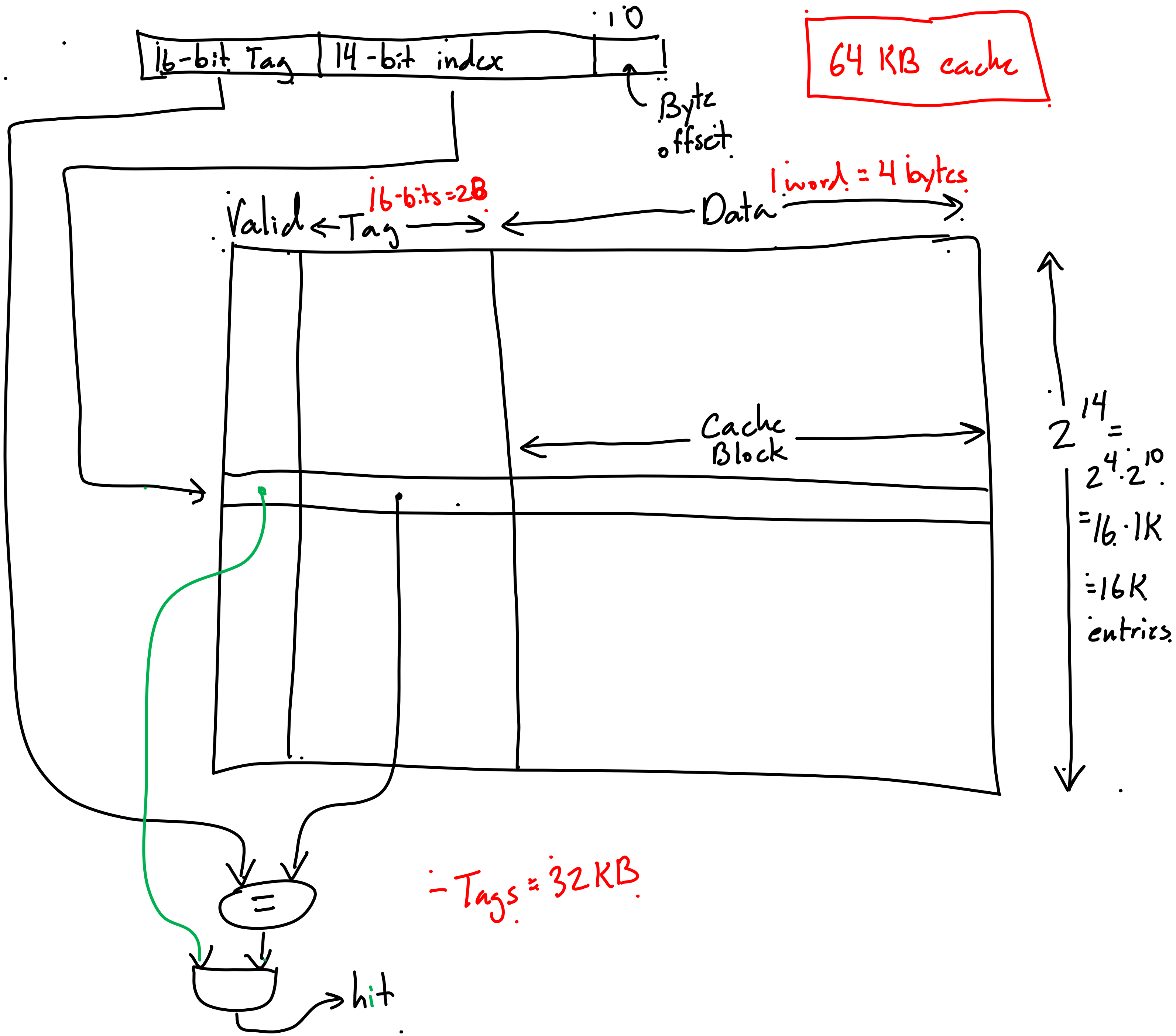
Hit Rate = $4/10$ = 40%

Miss Rate = 60%

- HW due Friday

Addressing into a cache



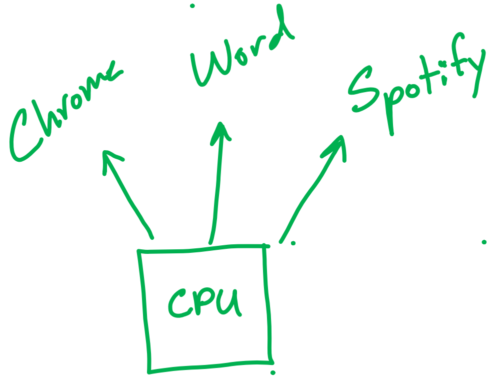


1 0
16-bit Tag
14-bit index
Byte offset
64 KB cache
1 word = 4 bytes
Valid
Tag
16-bits = 2B
Data
Cache Block
$2^{14} = 2^4 \cdot 2^{10} = 16 \cdot 1K = 16K$ entries
=
hit
- Tags = 32KB

Valid bit

2 Reasons:

1. On power up, the CPU clears all valid bits

2. 

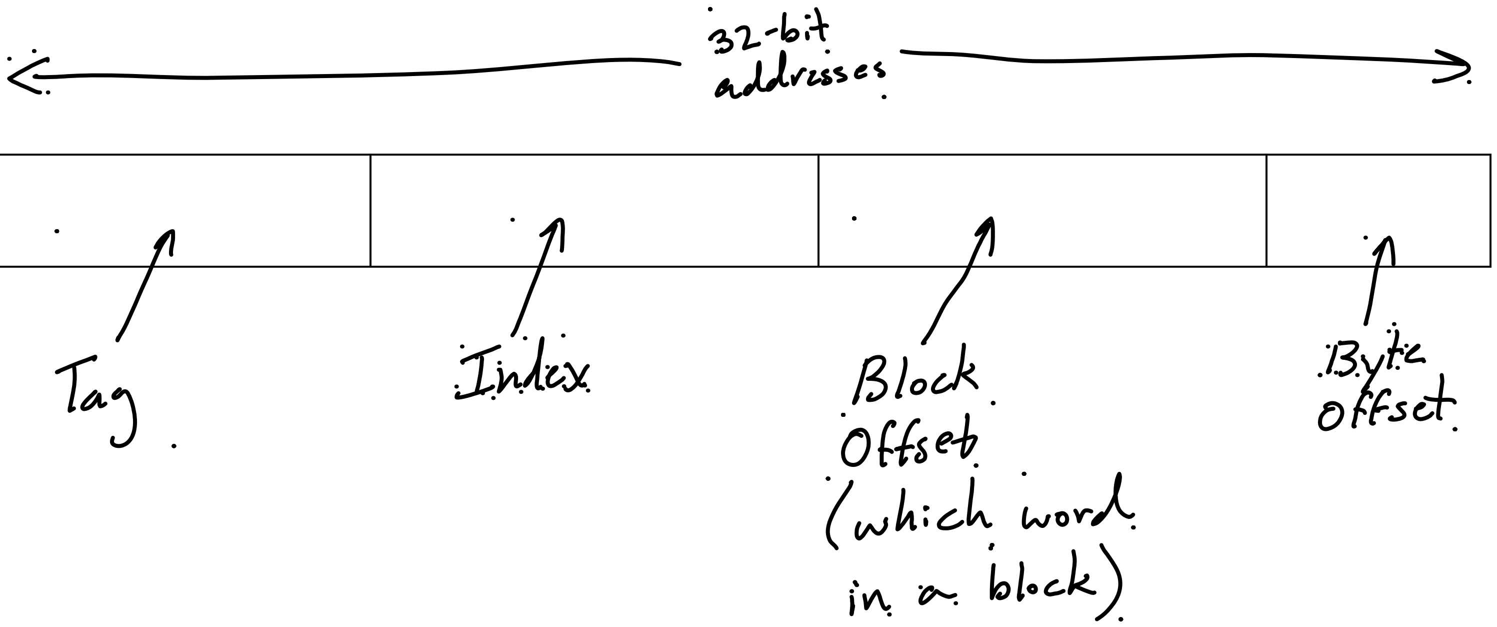
- When a context switch, the CPU can invalidate the cache

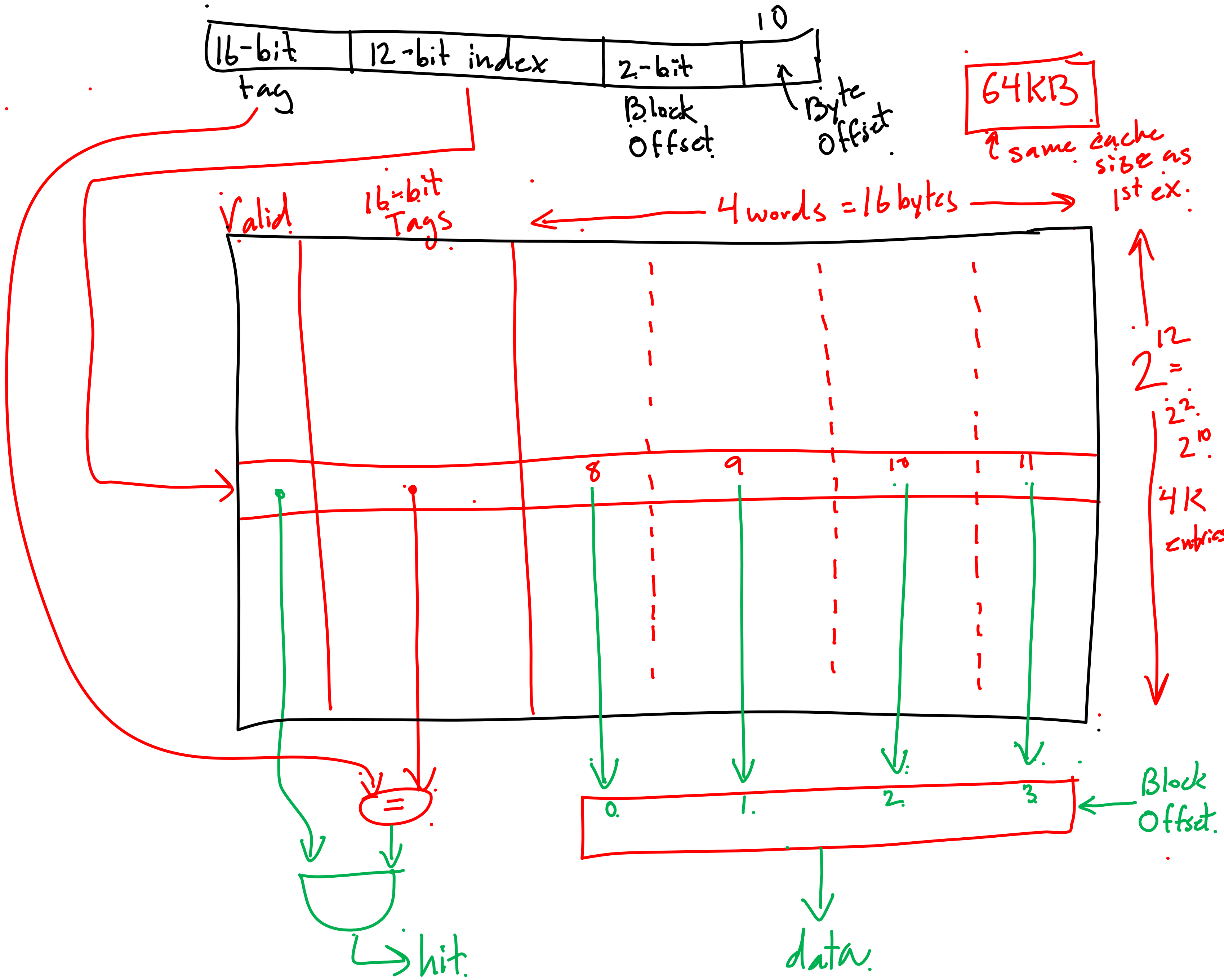
switching programs

Taking advantage of spatial locality

Multi-Word Cache Block - store neighboring words in the same block

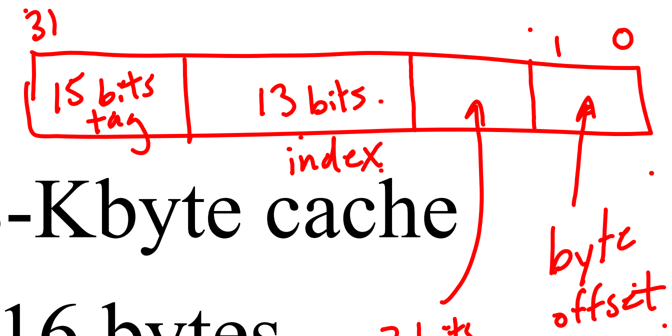
Address with multi-word blocks





16-bit tag
12-bit index
2-bit Block Offset
10
Byte Offset
64KB
same cache size as 1st ex.
Valid
16-bit Tags
4 words = 16 bytes
$2^{12} = 2^2 \cdot 2^{10}$
4K entries
8
9
10
11
0.
1.
2.
3
Block Offset
=
hit
data

Example

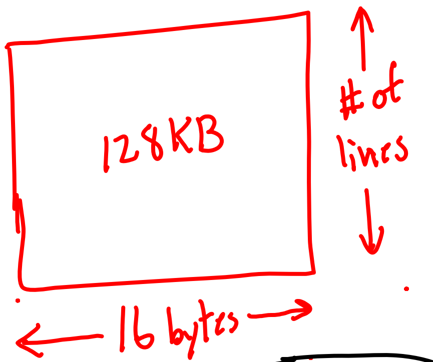


- You are implementing a 128-Kbyte cache
- The block size (line size) is 16 bytes.
- Each word is 4 bytes, address 32 bits
- How many bits is the block offset?

line size = 16 bytes = 4 words $\leftarrow 2^{\boxed{2}} \rightarrow$ $\boxed{\text{2 bits block offset}}$

- How many bits is the index?

$$\frac{\overset{128}{2^7} \cdot \overset{1k}{2^{10}}}{\underset{16}{2^4}} = 2^{\boxed{13}} \rightarrow \boxed{\text{13 bits for index}}$$



- How many bits is the tag?

$32 - 13 - 2 - 2 = \boxed{\text{15 bit tag}}$

Total # bits for tags $= \boxed{15 \cdot 2^{13} \text{ bits}}$

- Lab 5

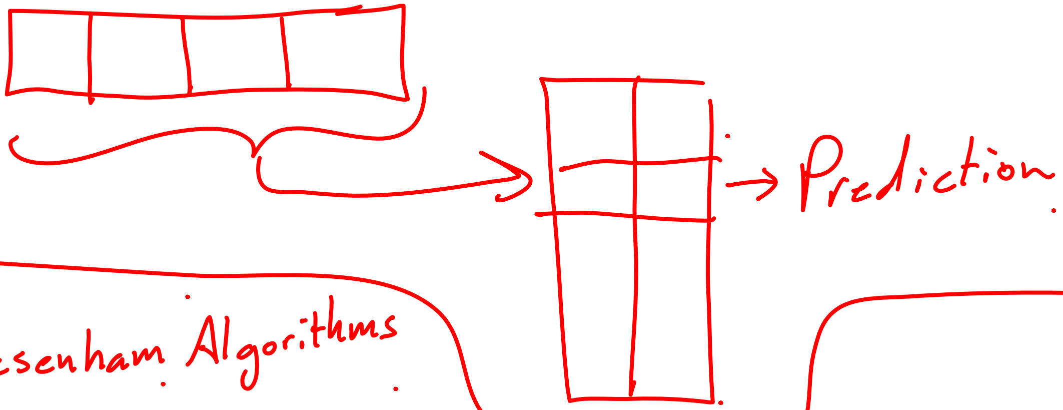
- Correlating Branch Predictor

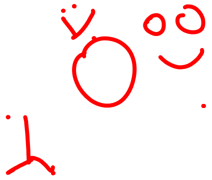
$$GHR(\overset{\overset{60+\%}{\uparrow}}{2}, 4, \overset{\overset{90+\%}{\uparrow}}{8})$$

mips > b

↑

branch predictor accuracy



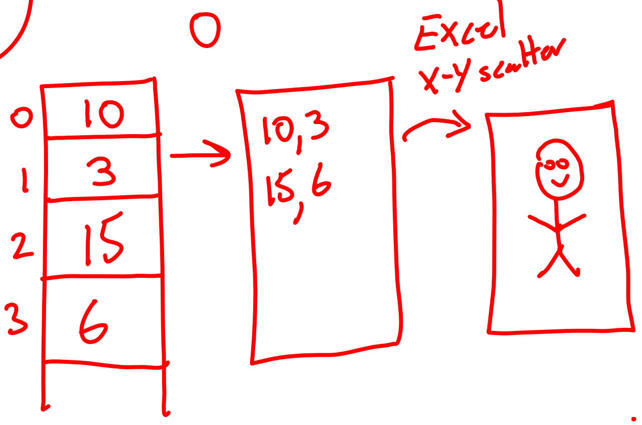


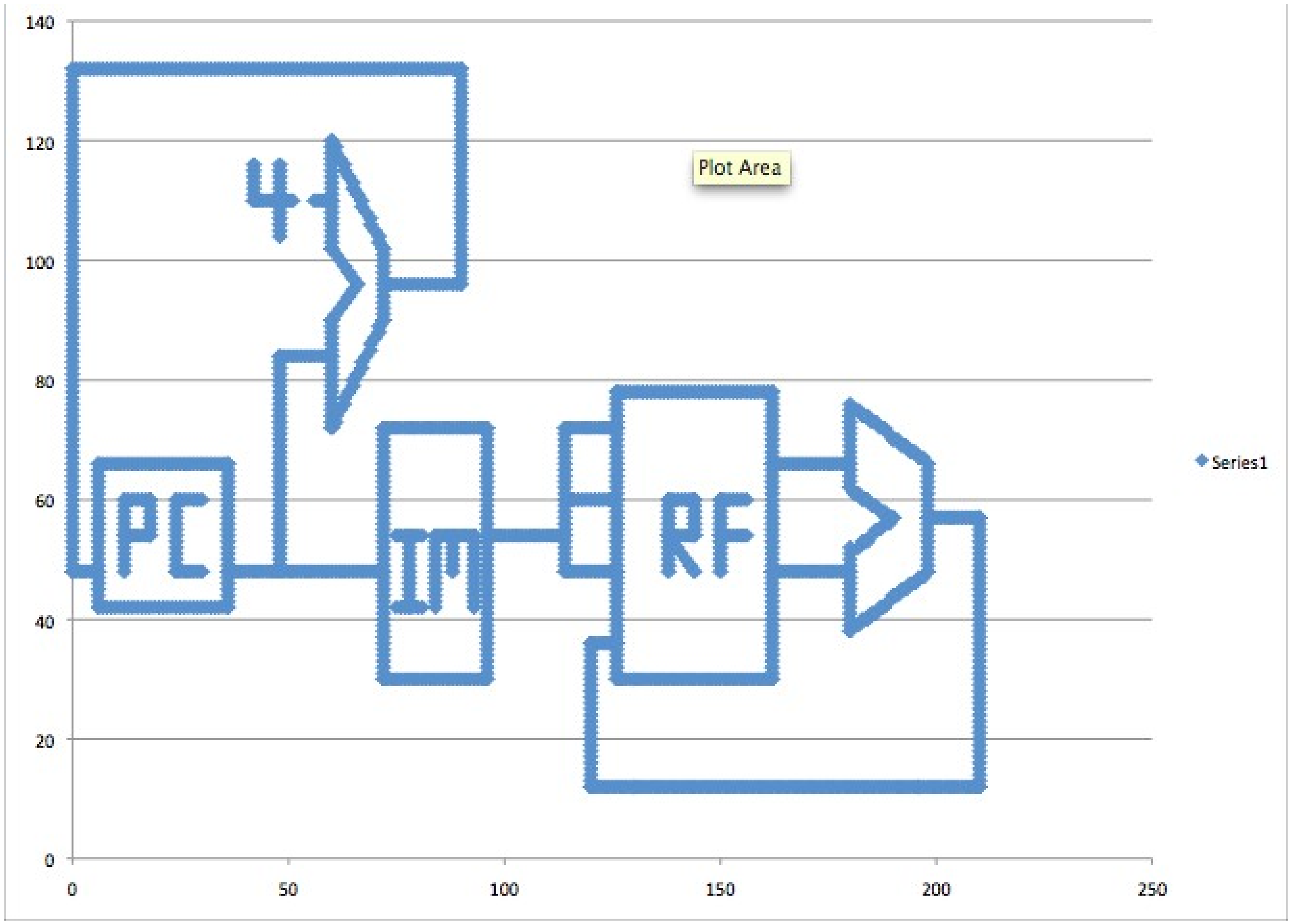
- Bresenham Algorithms

lines

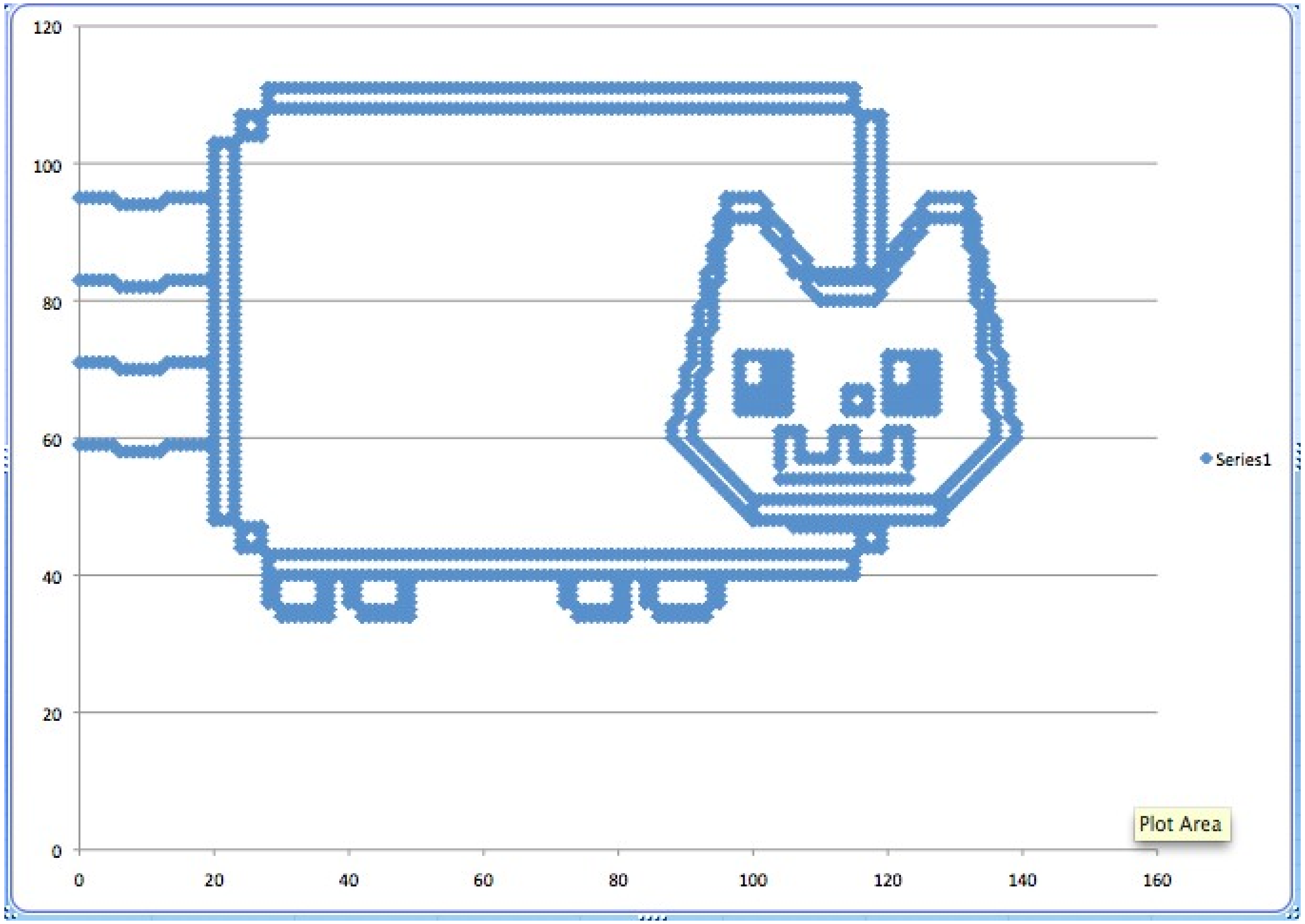
circles

figure.asm → plot(10,3)
plot(15,6)





140
120
100
80
60
40
20
0
0
50
100
150
200
250
Plot Area
4
PC
IM
RF
◆ Series1



120
100
80
60
40
20
0
0
20
40
60
80
100
120
140
160
● Series1
Plot Area