

# Memory (1A)

---

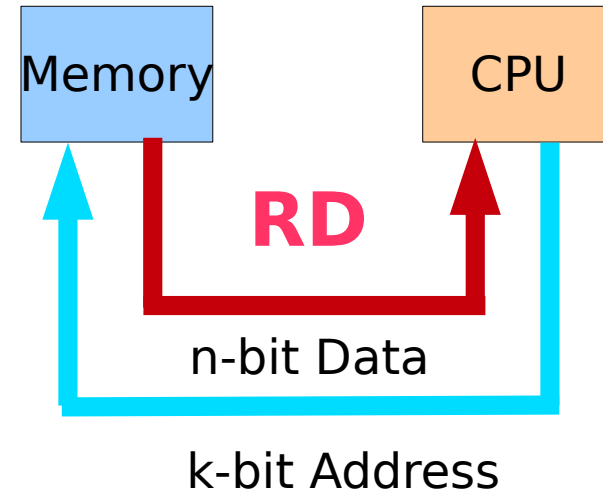
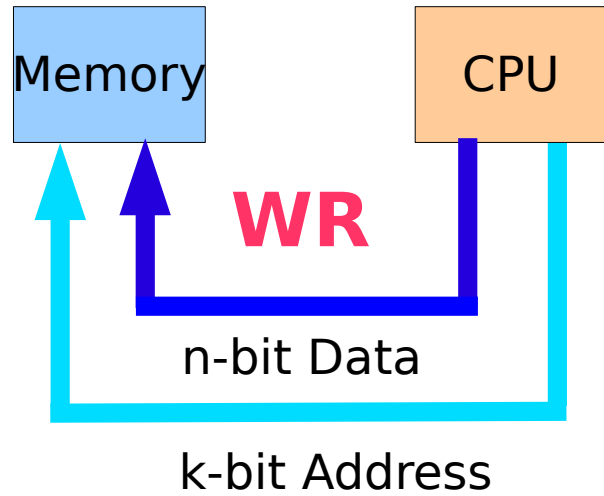
Copyright (c) 2011 - 2016 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using OpenOffice.

# Memory Access Operations

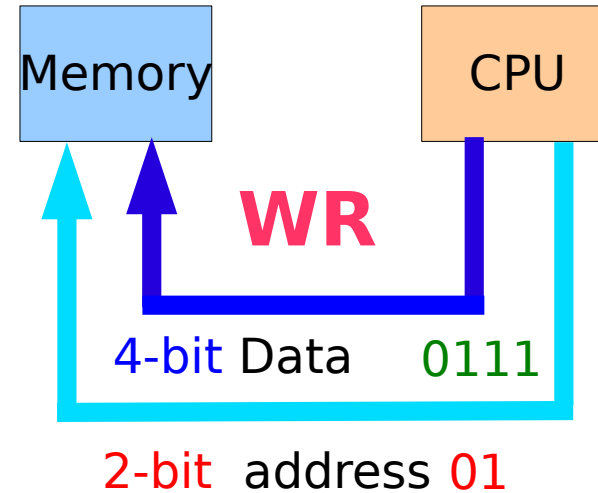


# A Memory Write Example

2-bit address	4-bit data
00	0101
01	
10	1100
11	



2-bit address	4-bit data
00	0101
01	0111
10	1100
11	



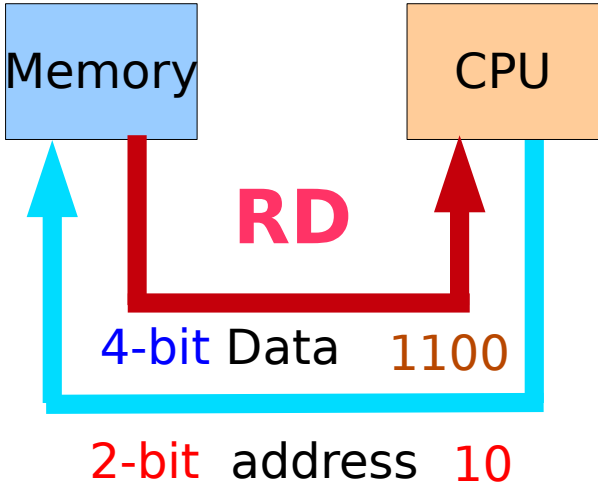
**WR**

Address: 01

Data: 0111

# A Memory Read Example

2-bit address	4-bit data
00	0101
01	0101
10	1100
11	

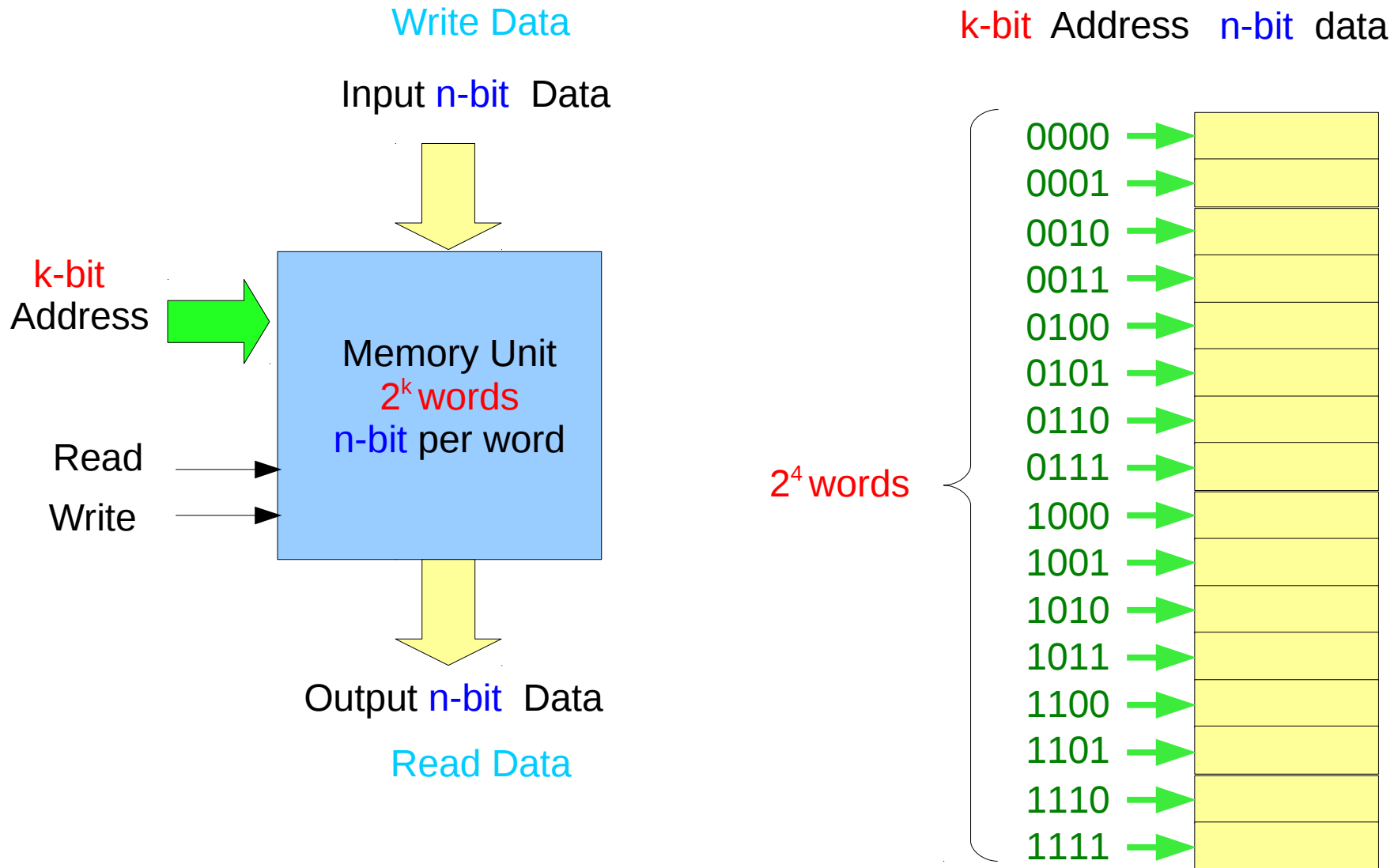


**RD**

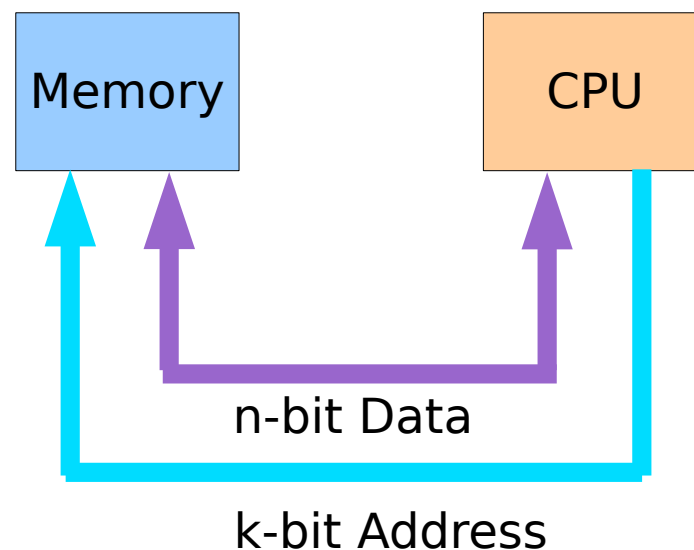
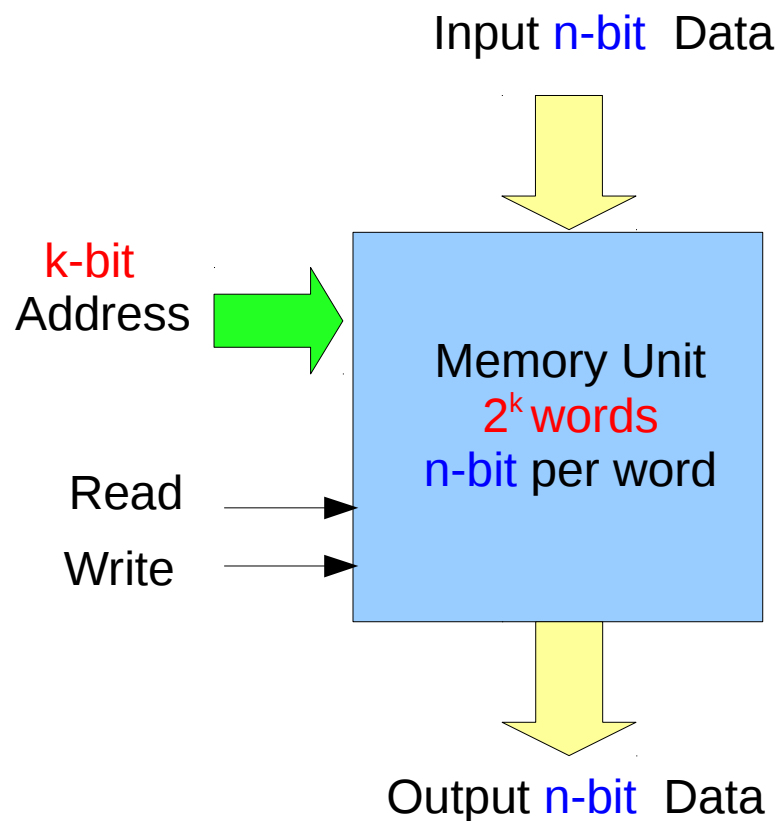
Address: 10

Data: → 1100

# A Memory and A Memory Map

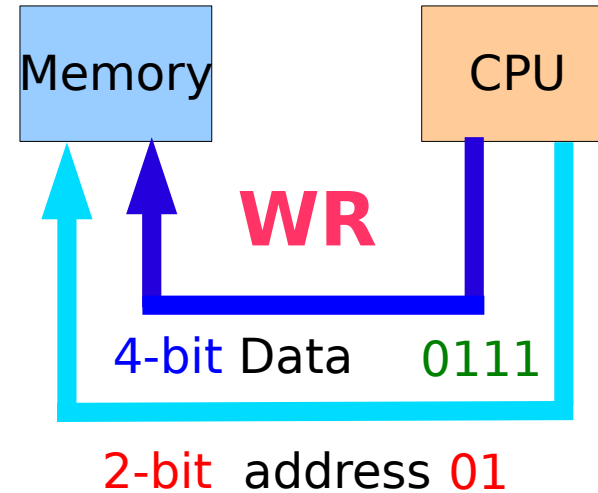


# The Inputs and Outputs of a Memory



# Assignment and Write Operation

2-bit address	4-bit data
00	0101
01	0111
10	1100
11	



address	data
&a	a

int a;

a = 7;

LHS (Left Hand Side)

**WR**

Address: 01

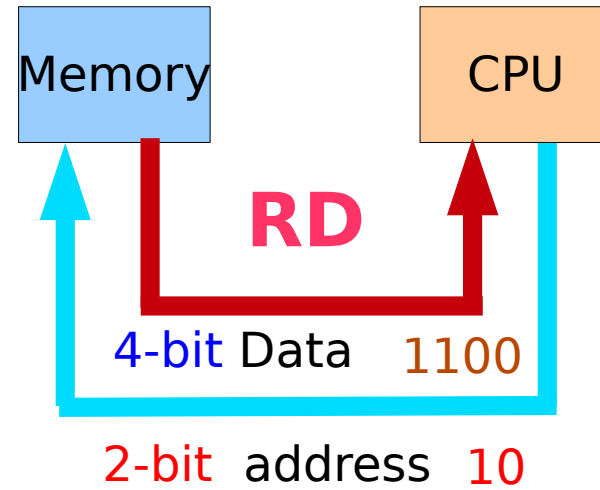
Data: 0111



# Assignment and Read Operation

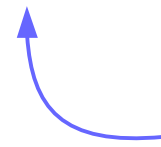
2-bit address	4-bit data
00	0101
01	0101
10	1100
11	

address	data
&a	a
&b	b



int a;

x = a + b;



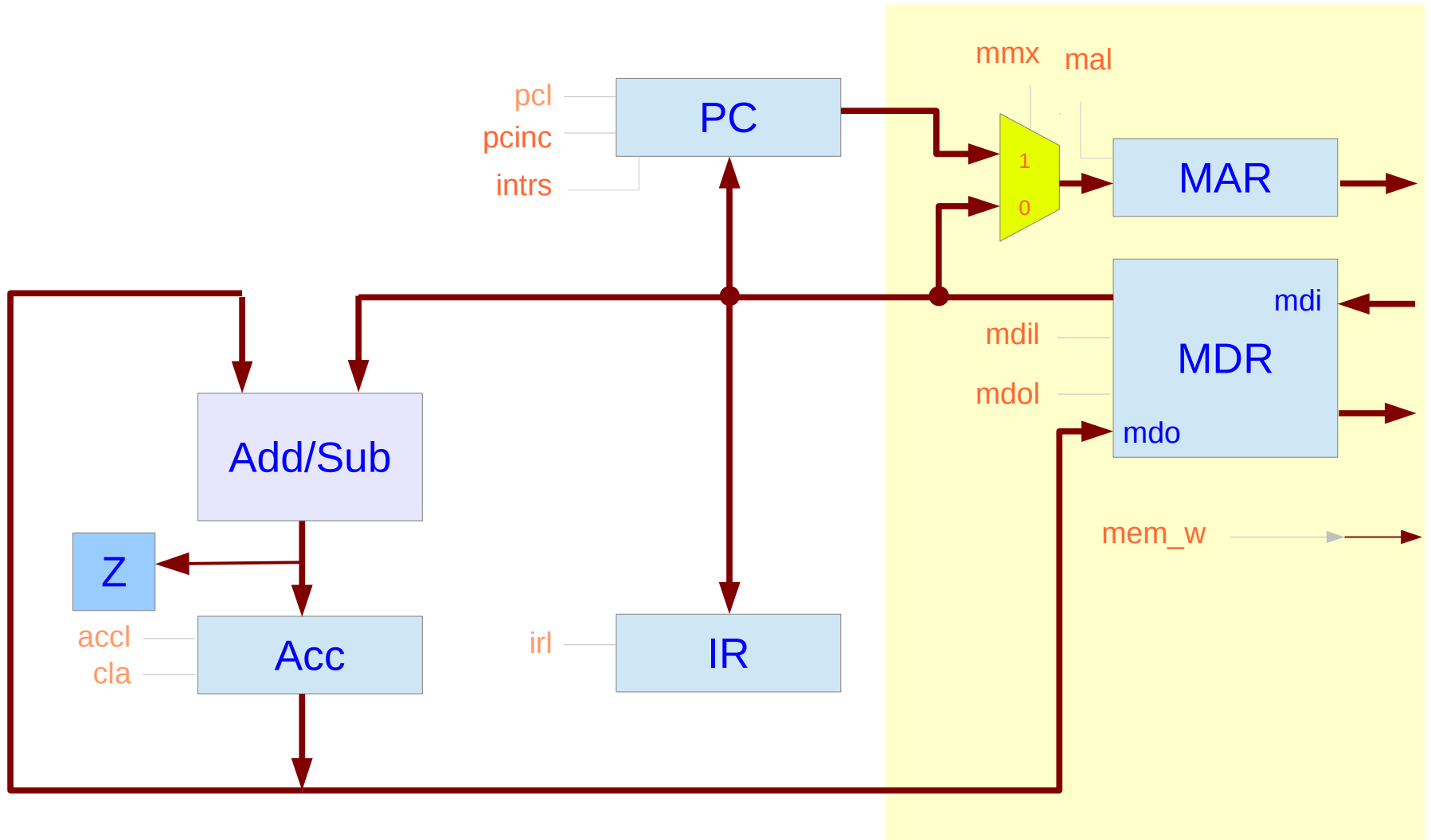
RHS (Right Hand Side)

**RD**

Address: 10

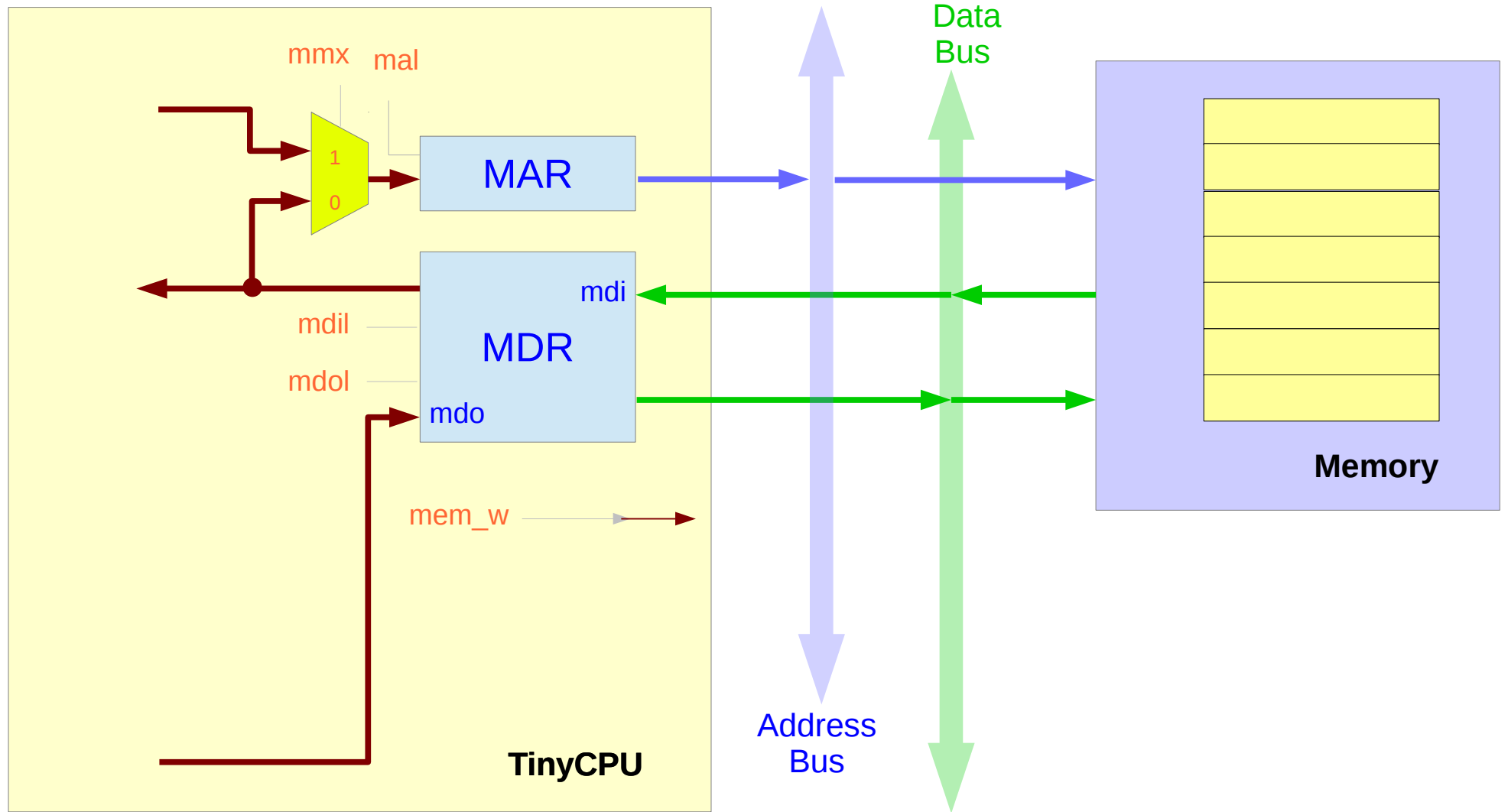
Data: → 1100

# TinyCPU



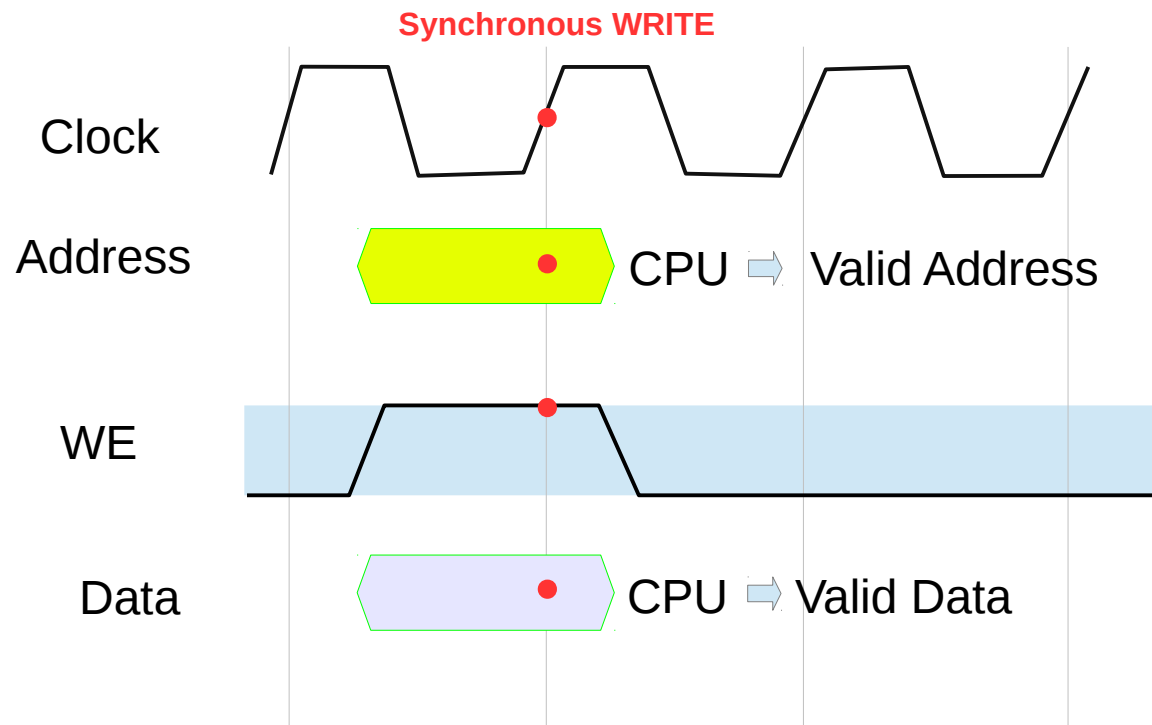
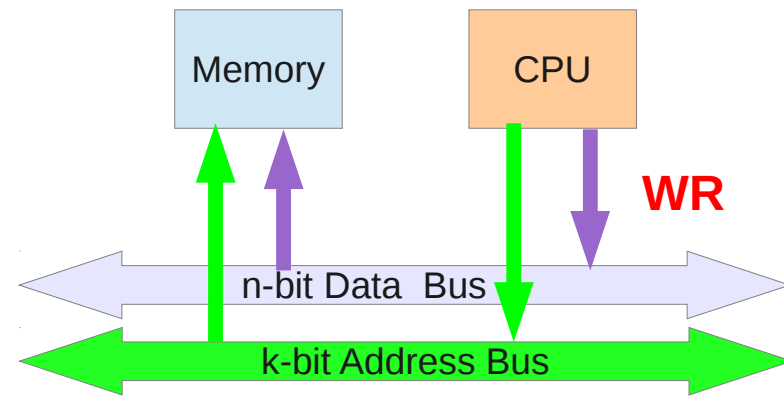
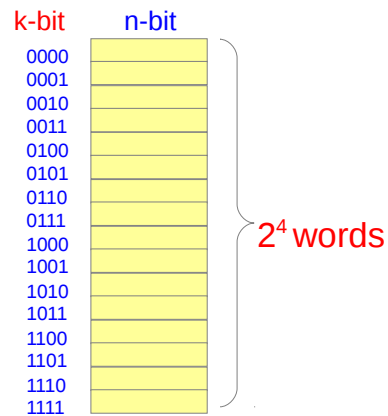
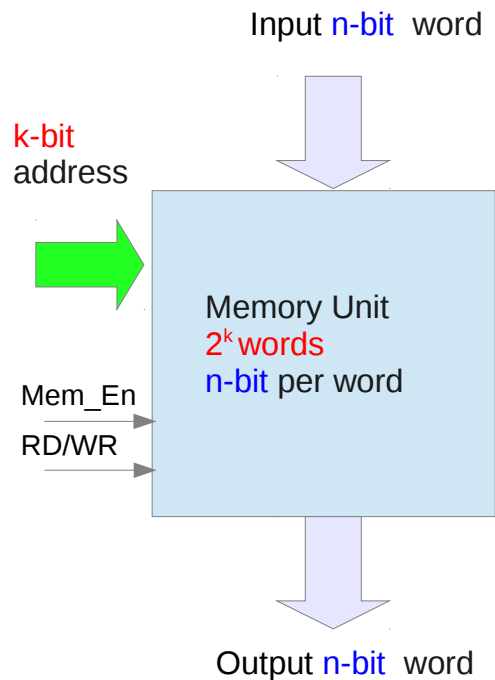
Based on <http://www.ele.uri.edu/Courses/ele306/f01/Tinydoc.pdf>

# TinyCPU and Memory

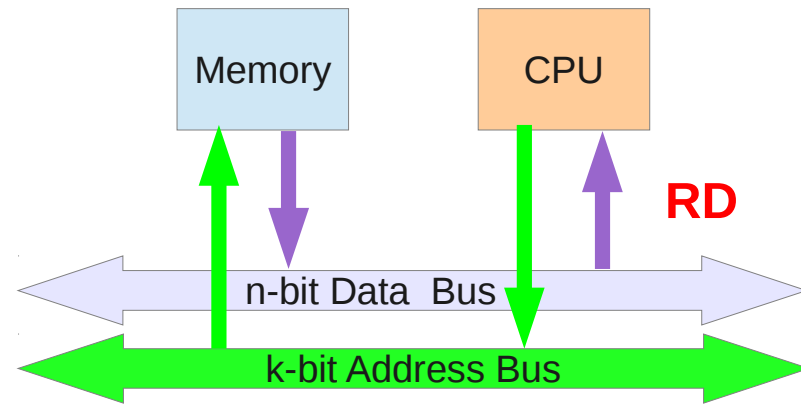
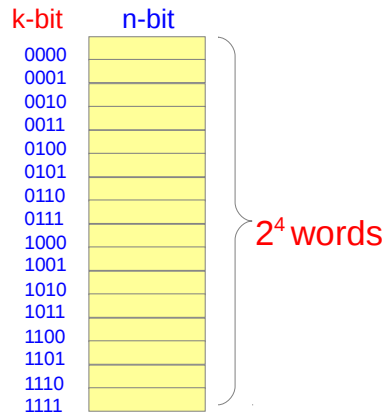
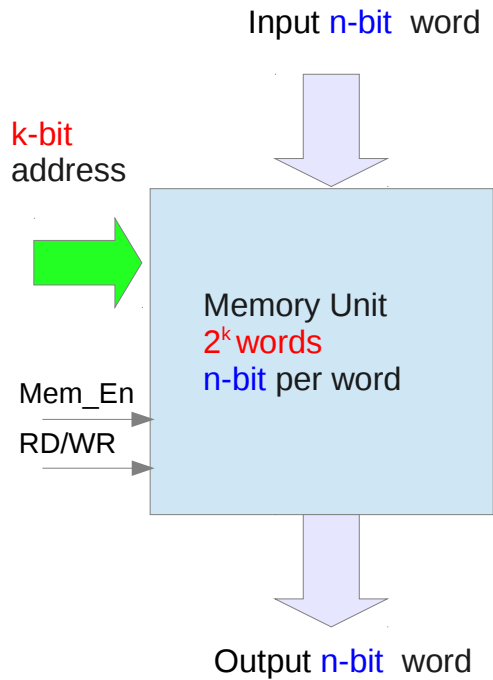


Based on <http://www.ele.uri.edu/Courses/ele306/f01/Tinydoc.pdf>

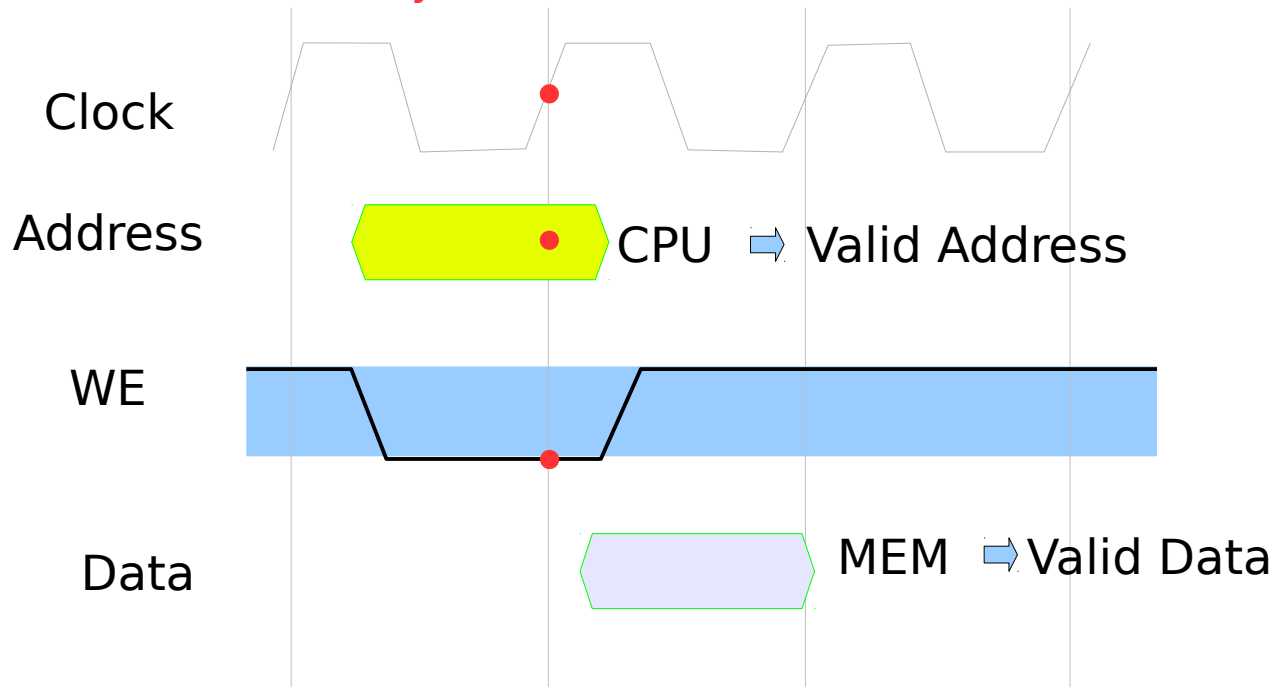
# Memory Write Cycle



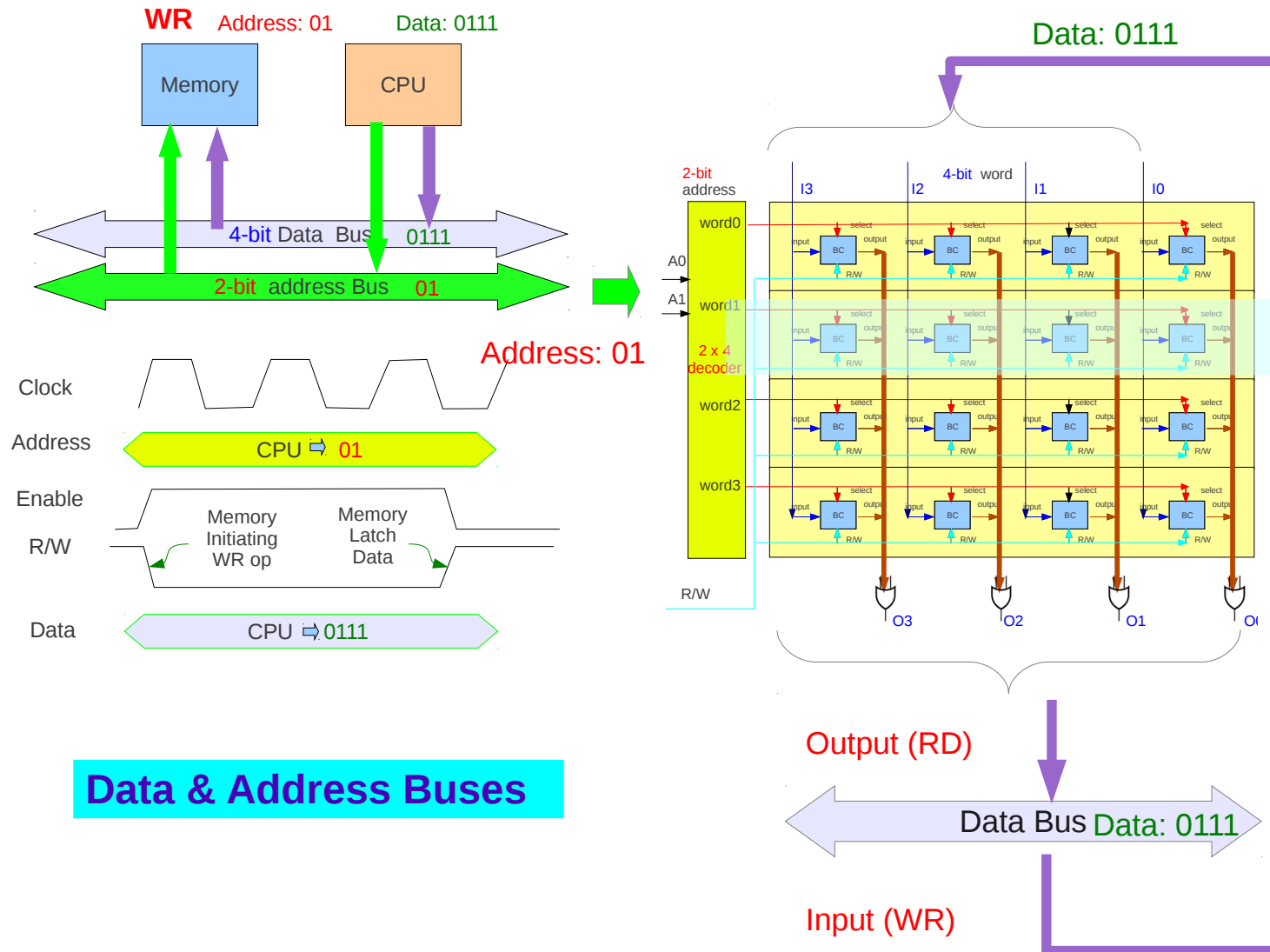
# Memory Read Cycle - Synchronous



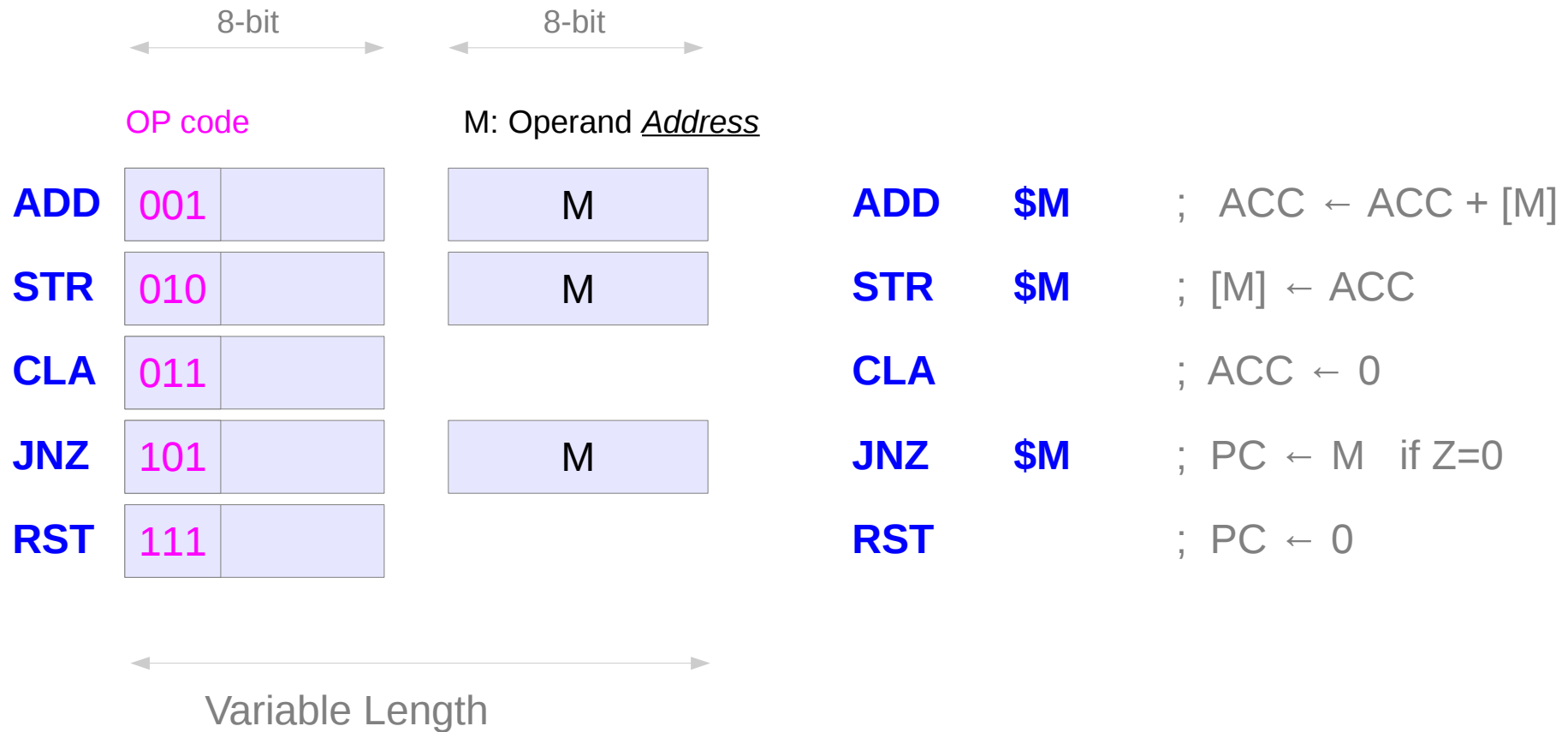
## Synchronous READ



# Write Cycle Example for a 4x4 Memory



# Instruction Set Architecture



Based on <http://www.ele.uri.edu/Courses/ele306/f01/Tinydoc.pdf>

### Addr. Machine Codes

00	60		CLA		;Clear Acc and then adding the operand is
01	20	78	ADD \$ONE		;equivalent to move the operand to Acc
03	40	FF	STR \$FF		
05	A0	01	JNZ \$01		
07	E0		RST		
78	01	ONE:	DAT	0000 0001	;Constant 1

00 → 01 → ... → FE → FF → 00

00	60
01	20
02	78
03	40
04	FF
05	A0
06	01
07	E0
78	01

FF	
----	--



# Array

---

## References

- [1] Essential C, Nick Parlante
- [2] Efficient C Programming, Mark A. Weiss
- [3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
- [4] C Language Express, I. K. Chun