Basic CPU – ISA (1A)

Young Won Lim
3/22/25

Based on 컴퓨터구조론 (개정 6 판), 김종현 , 생능출판사

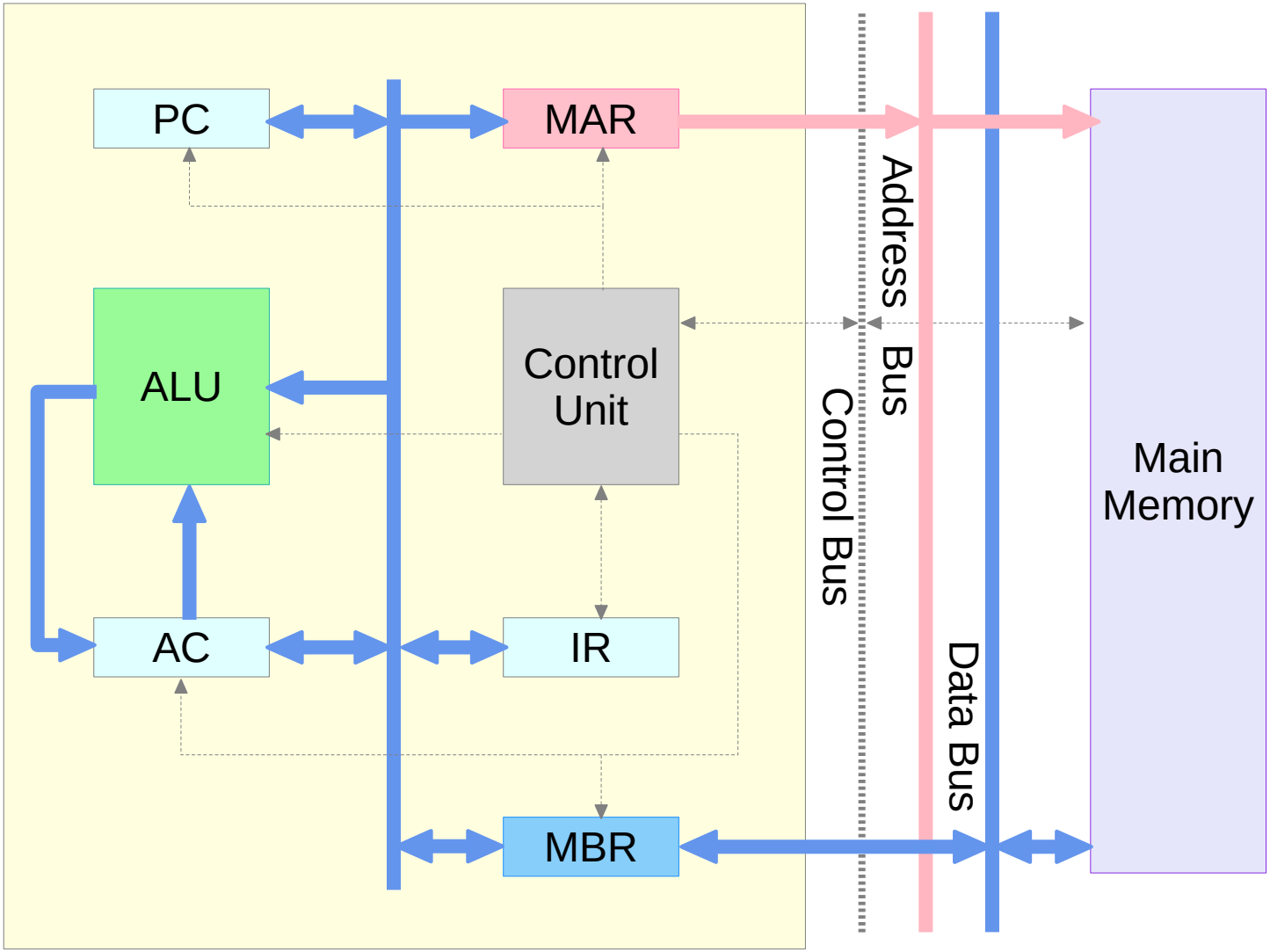
(Computer Architecture, 6th ed, Jonghyun Kim, Life & Power Press, Korea)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

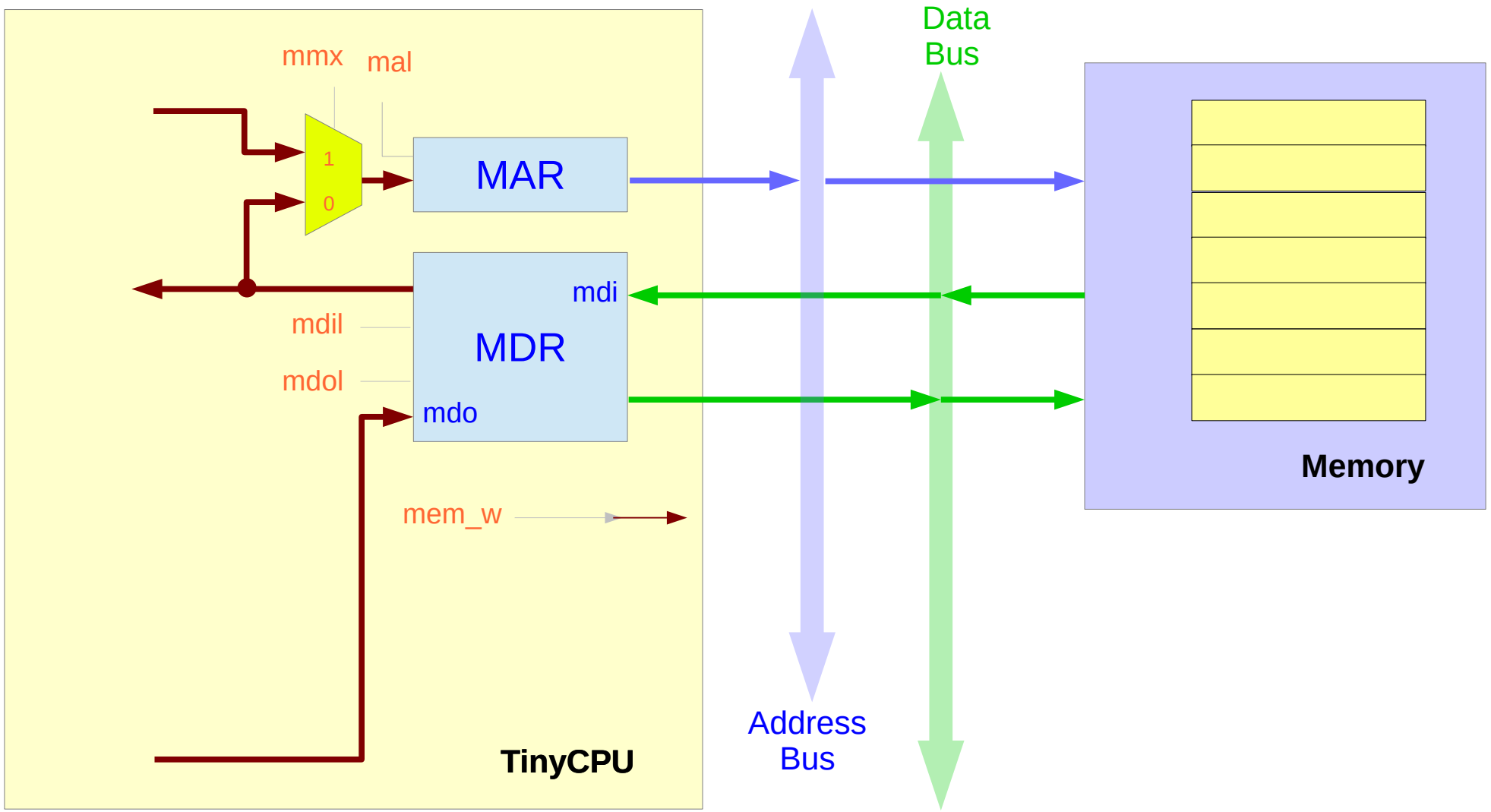
This document was produced by using LibreOffice and Octave.

Data Path

Basic CPU

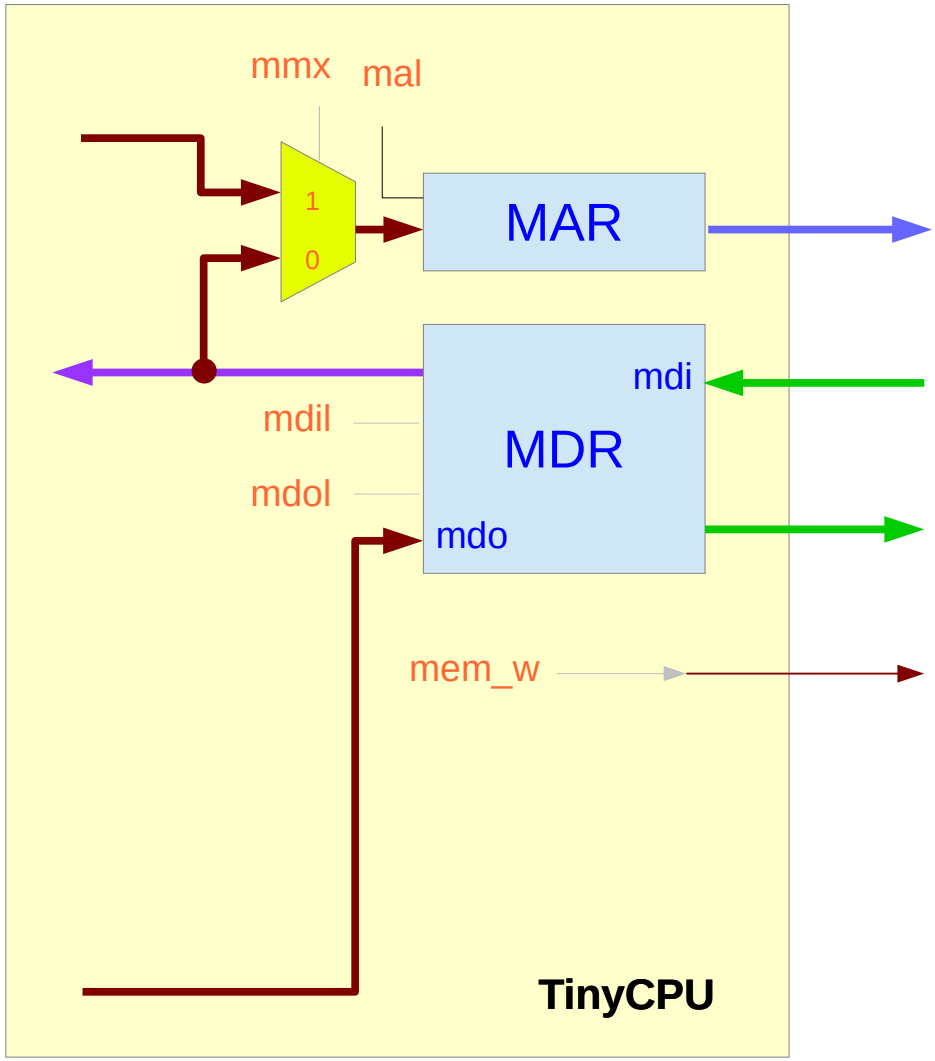


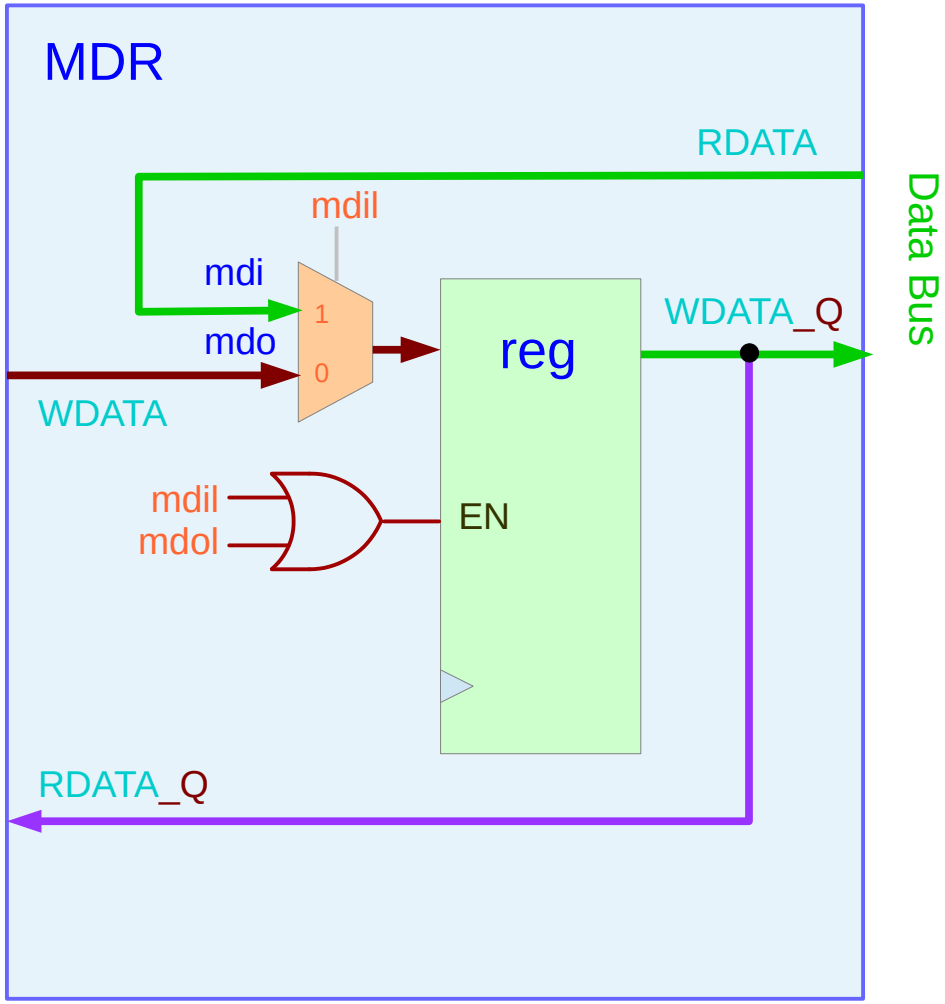
CPU and MEM



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

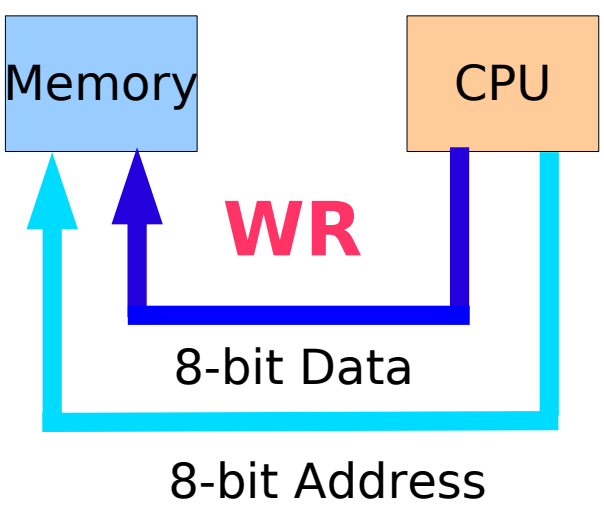
Memory Data Register

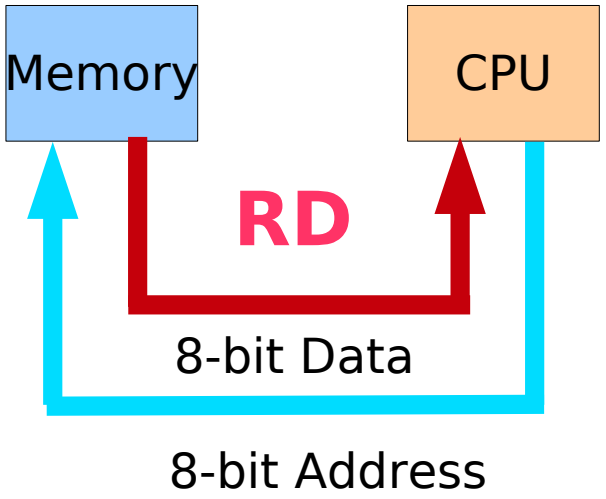




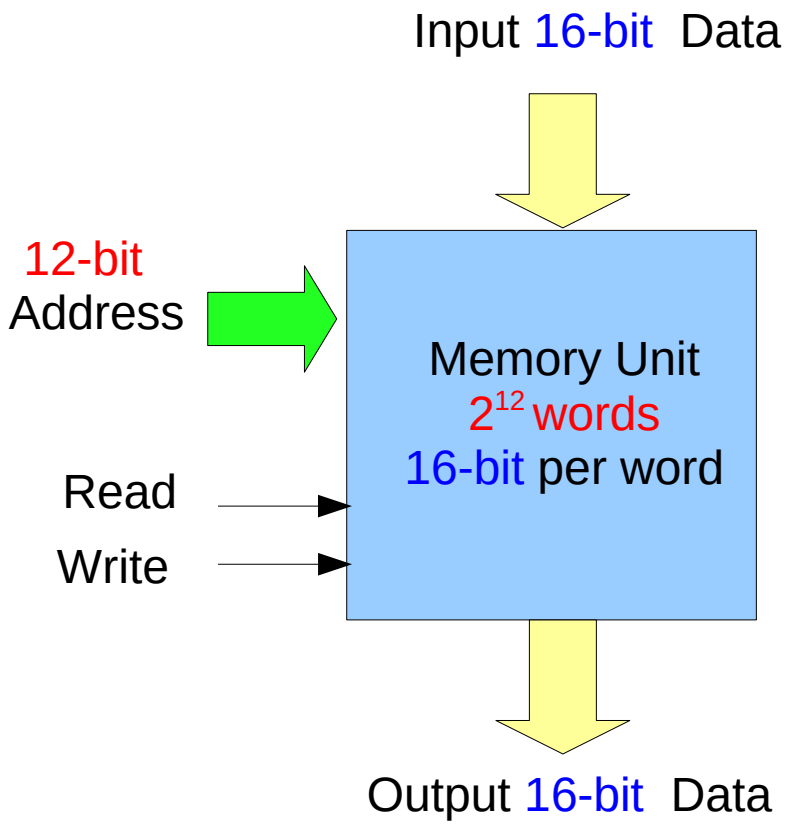
Based on http://www.ele.uri.edu/Courses/ele306/f01/Tinydoc.pdf

Memory Access Operations

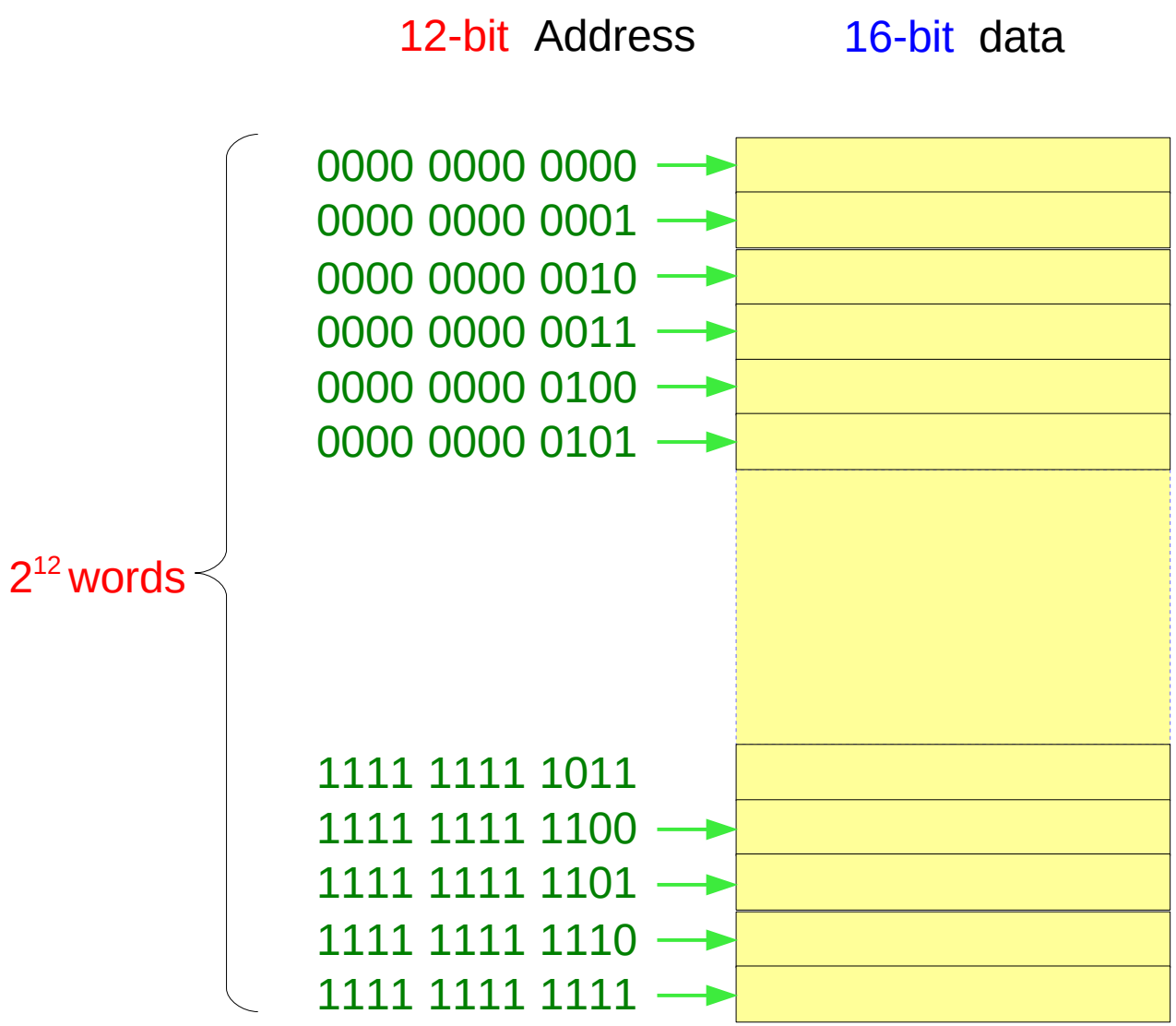




Memory Interface



Memory Map


12-bit Address
16-bit data
0000 0000 0000
0000 0000 0001
0000 0000 0010
0000 0000 0011
0000 0000 0100
0000 0000 0101
2^12 words
1111 1111 1011
1111 1111 1100
1111 1111 1101
1111 1111 1110
1111 1111 1111

Instruction Set Architecture

LOAD ***addr*** ; AC ← M[*addr*]

STA ***addr*** ; M[*addr*] ← AC

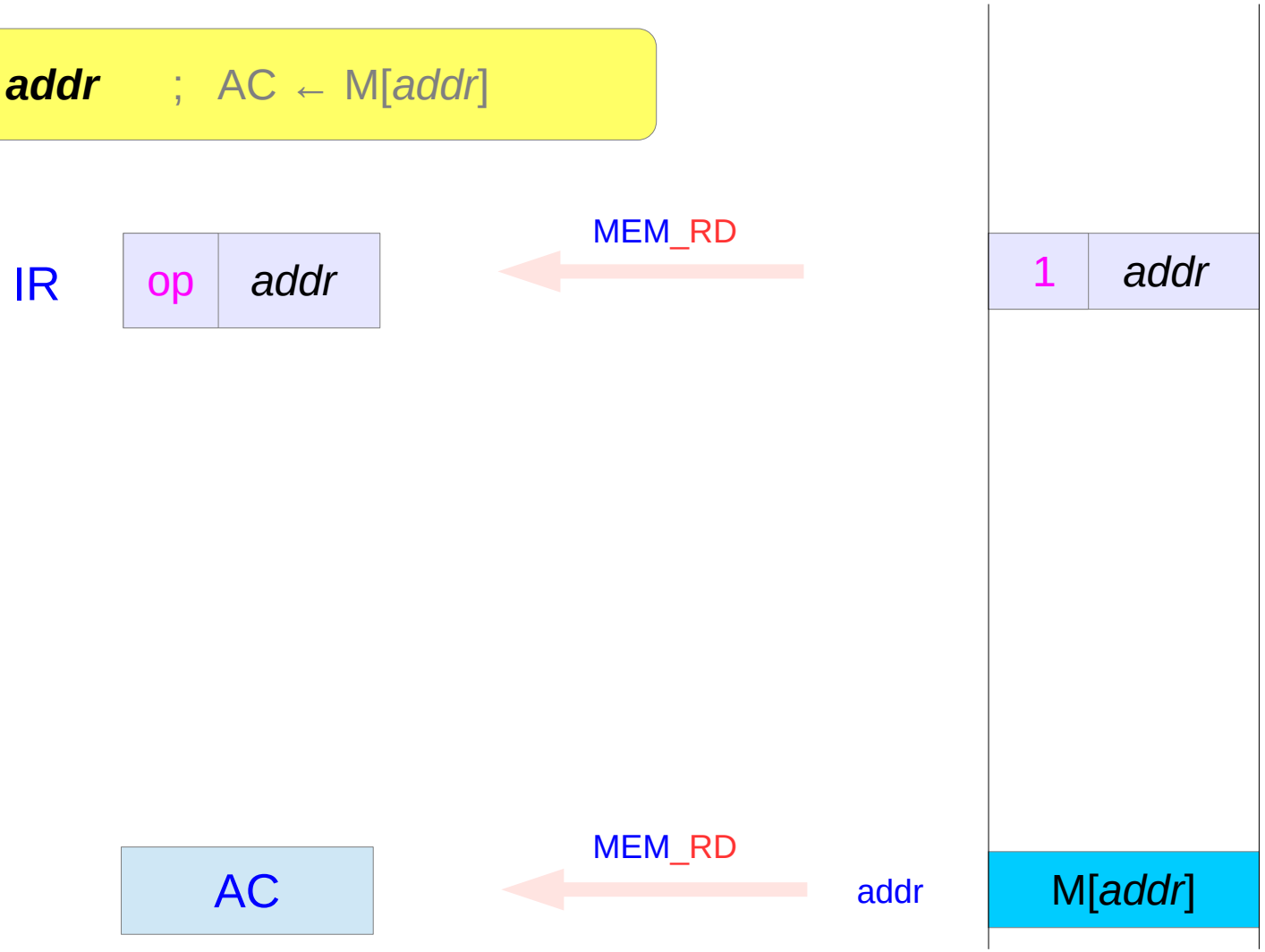
ADD ***addr*** ; AC ← AC + M[*addr*]

JUMP ***addr*** ; PC ← *addr*

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

LOAD Operation

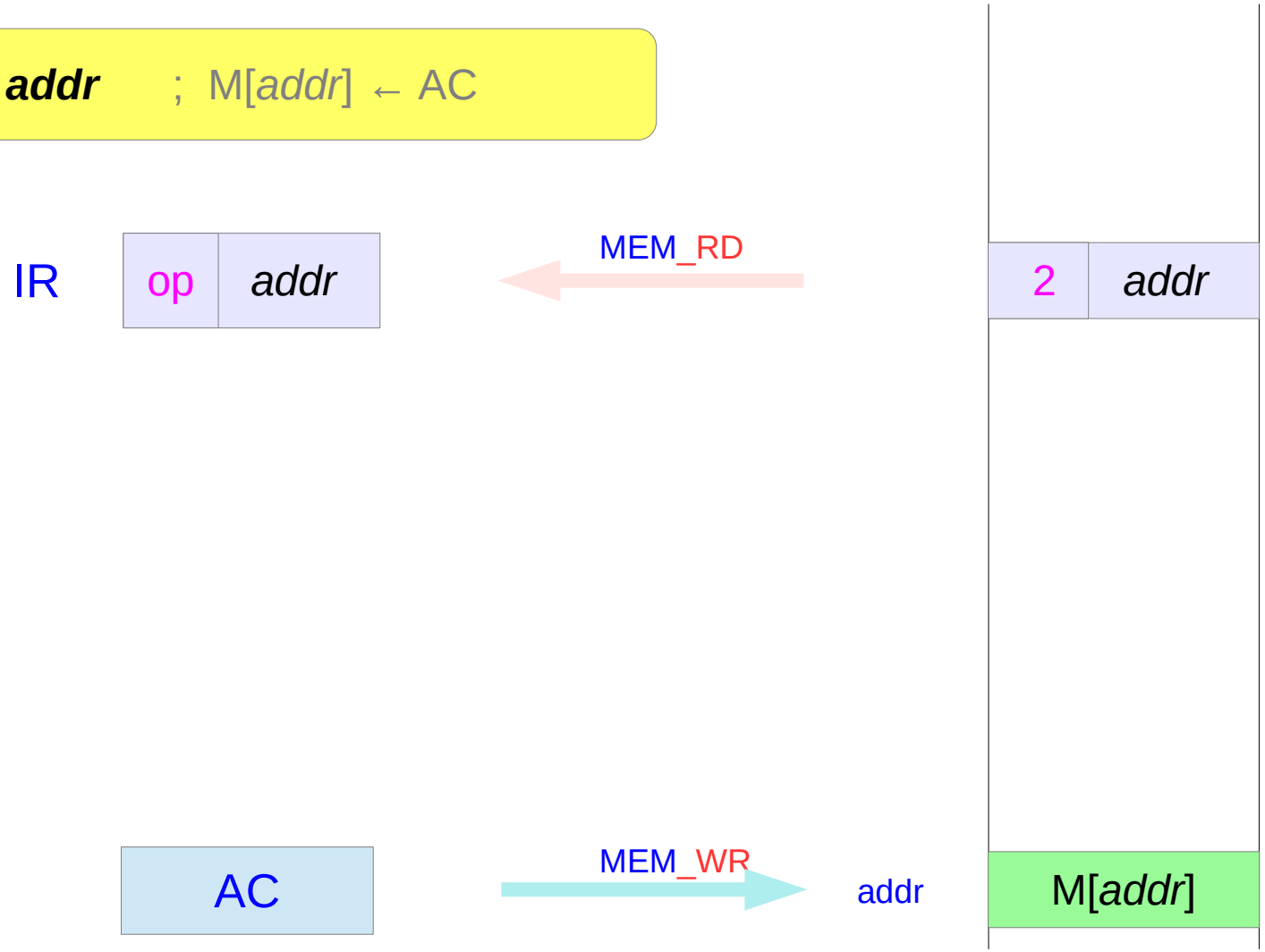
LOAD ***addr*** ; AC ← M[*addr*]



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

STA Operation

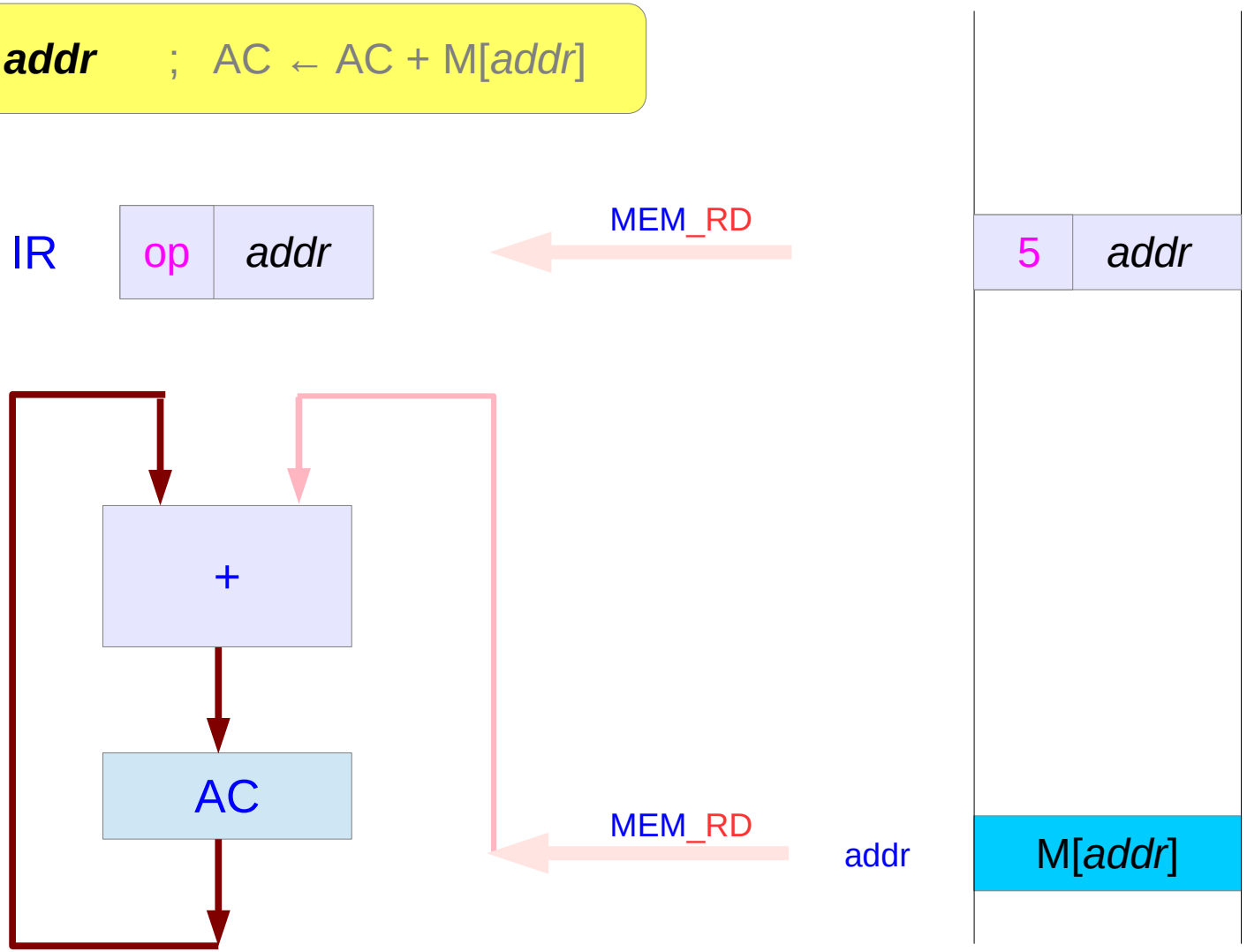
STA ***addr*** ; M[*addr*] ← AC



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

ADD Operation

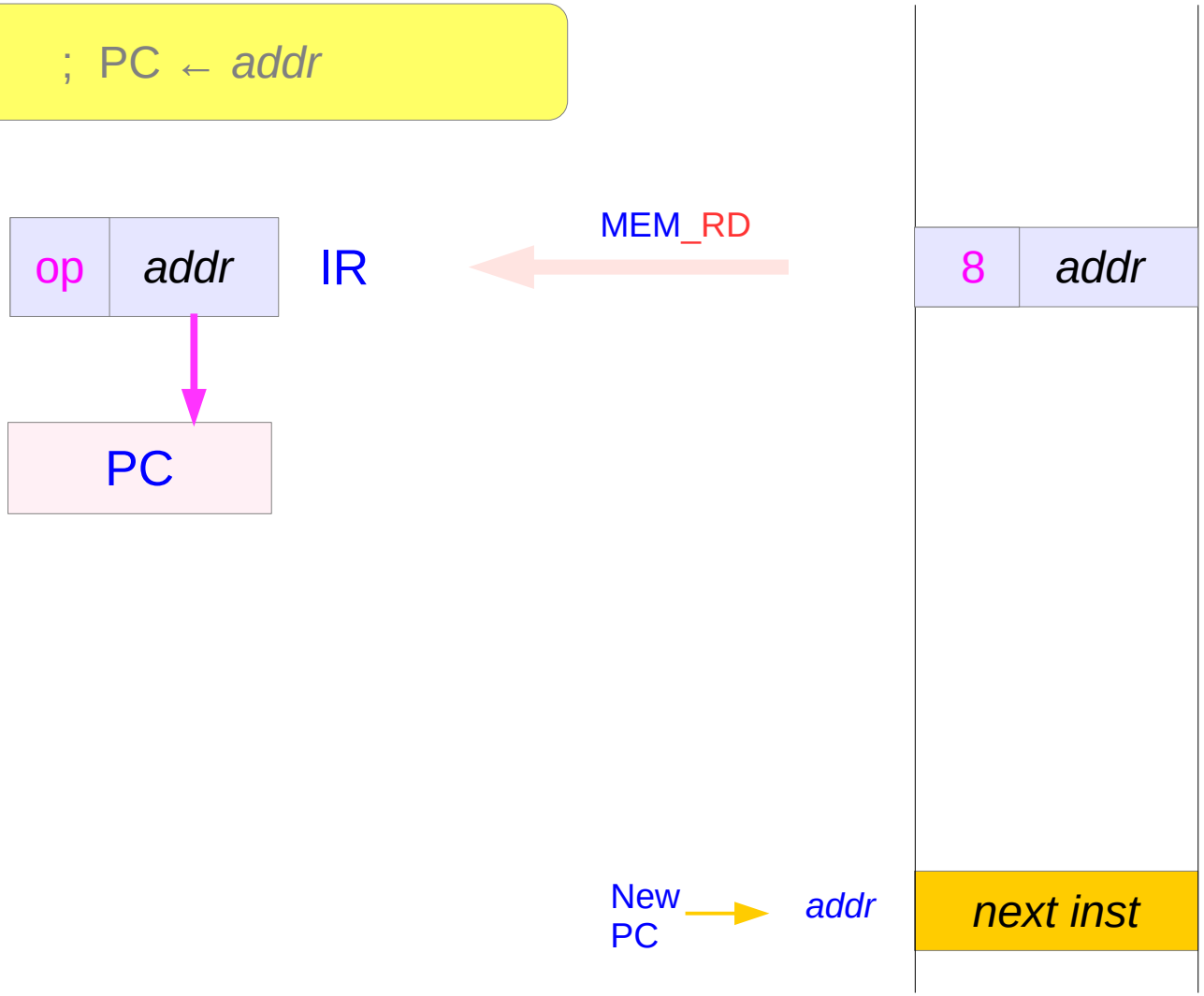
ADD ***addr*** ; AC ← AC + M[*addr*]



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

JUMP Operation

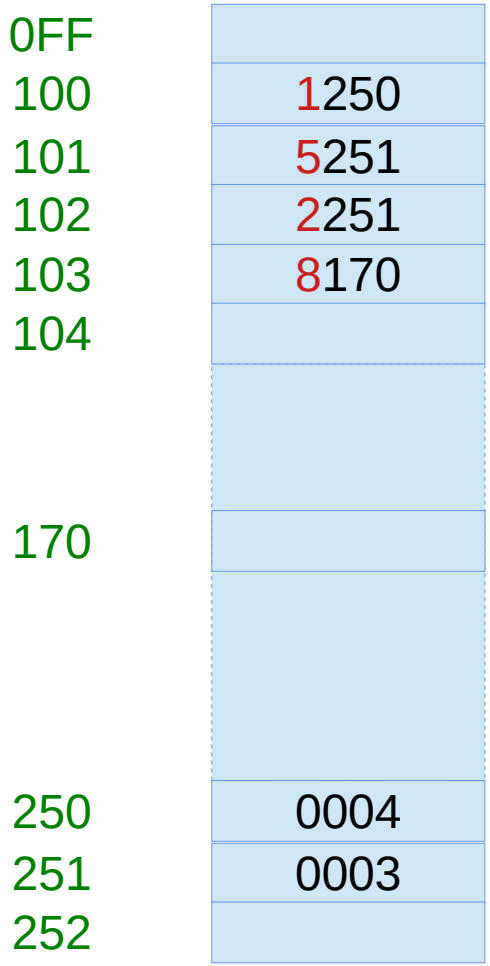
JUMP ***addr*** ; PC ← *addr*



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

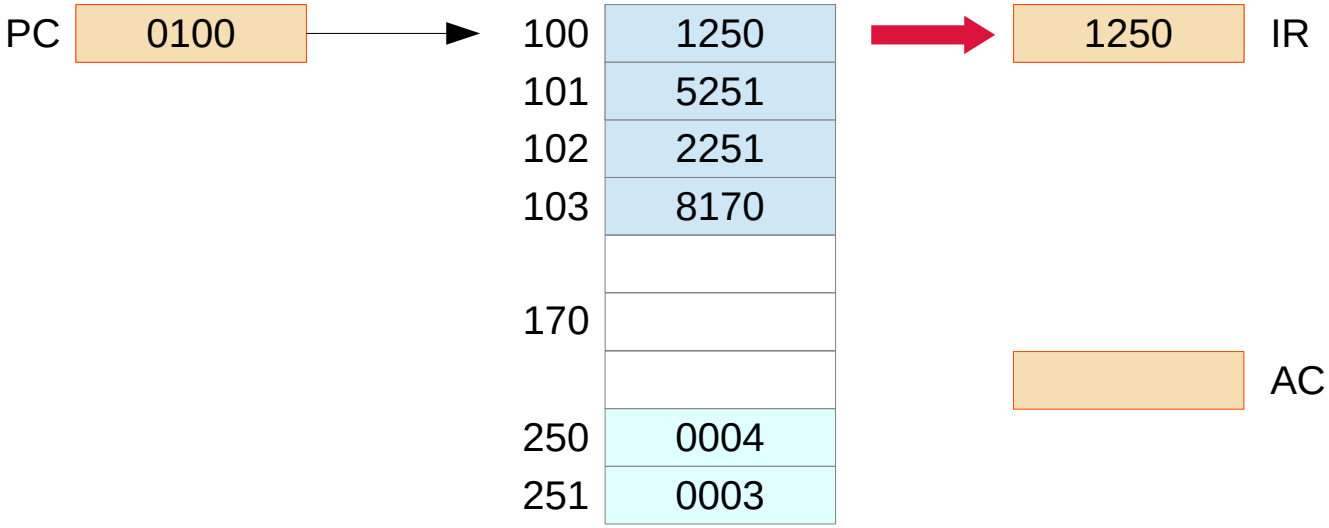
An Example Program

12-bit Address 16-bit data



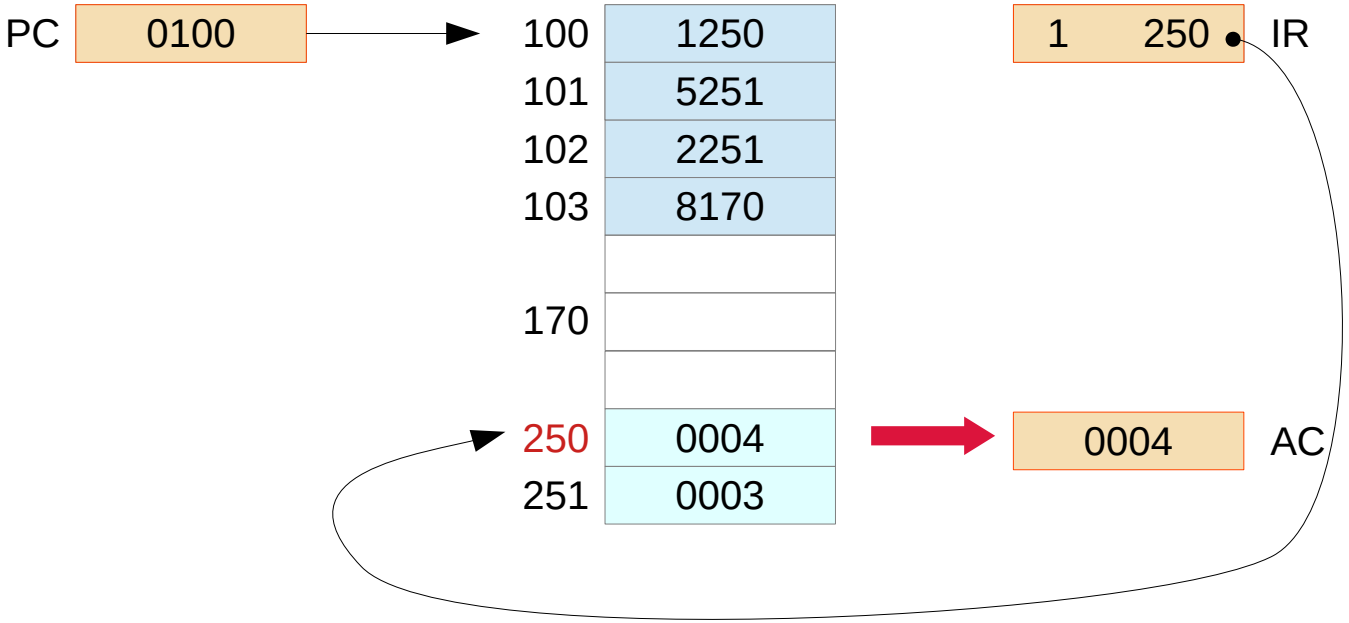

```
LOAD   250    ;  AC ← M[250]
STA    251    ;  M[251] ← AC
ADD    251    ;  AC ← AC + M[251]
JUMP   170    ;  PC ← 170
```

The 1st Instruction : LOAD 250 – Fetch Cycle



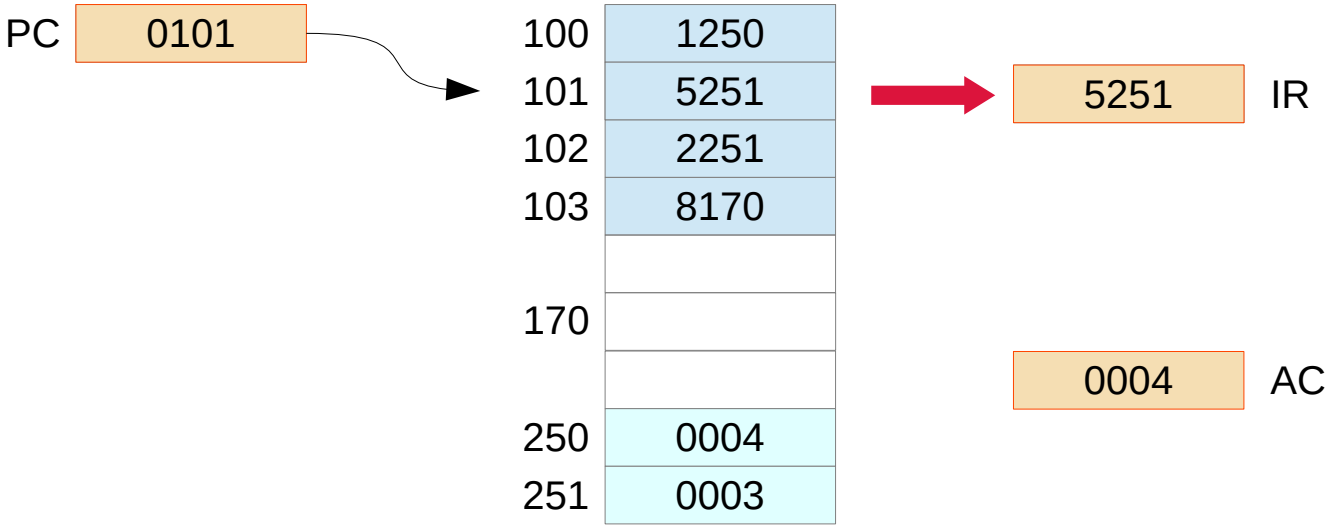
LOAD ***250*** ; AC ← M[*250*]

The 1st Instruction : LOAD 250 – Execution Cycle



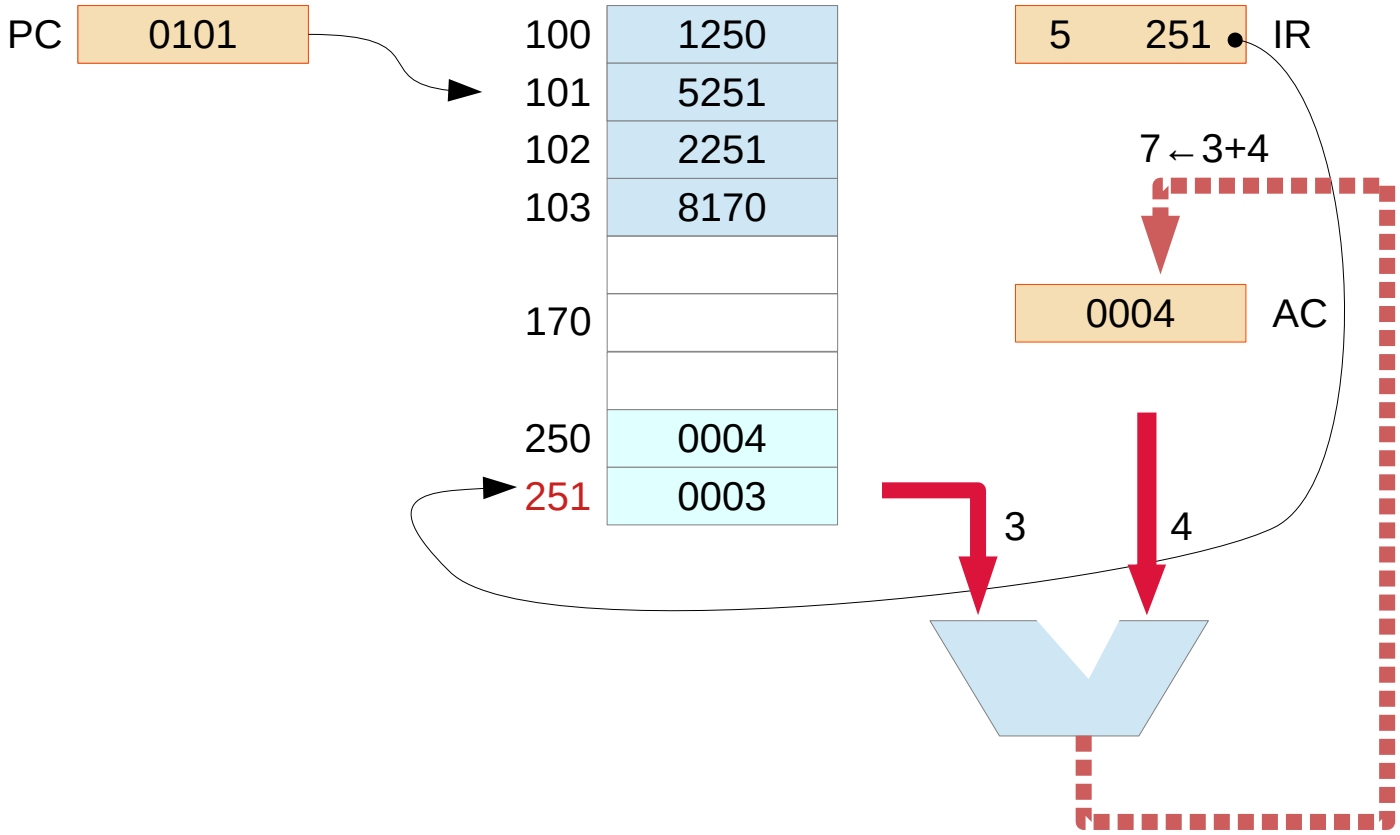
LOAD ***250*** ; AC ← M[*250*]

The 2nd Instruction : ADD 251 – Fetch Cycle



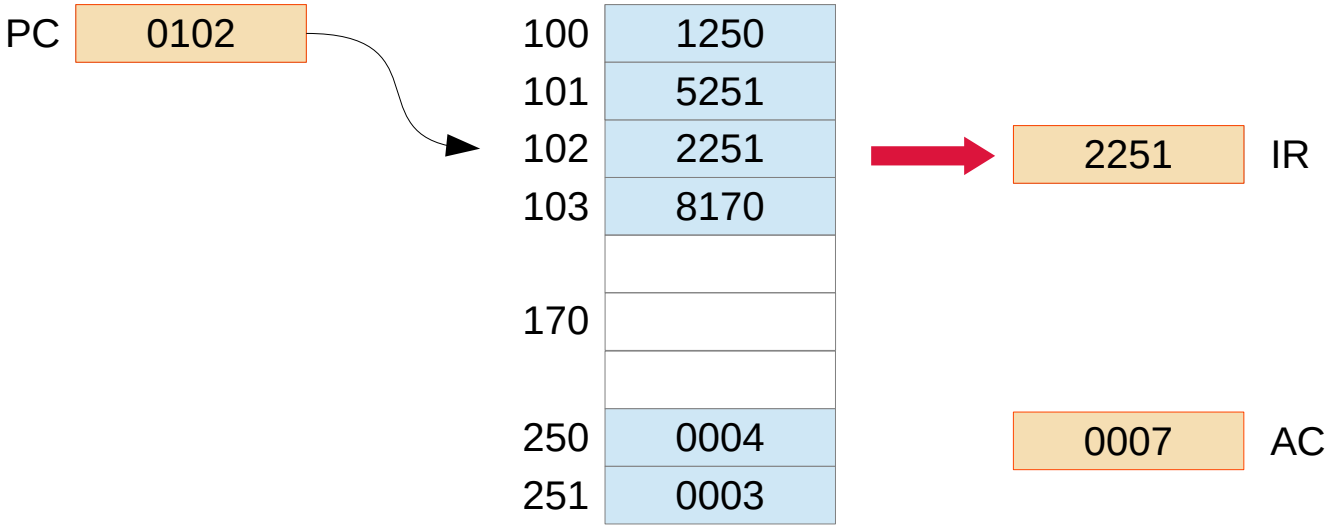
ADD ***251*** ; AC ← AC + M[*251*]

The 2nd Instruction : ADD 251 – Execution Cycle



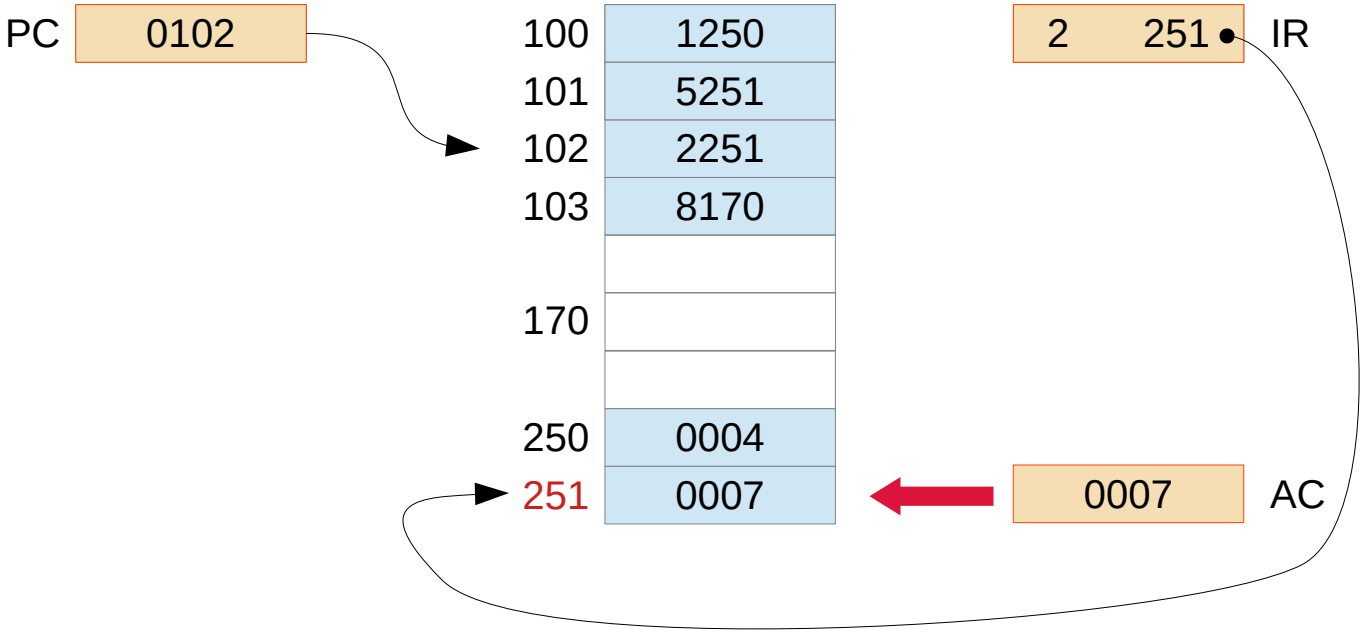
ADD ***251*** ; AC ← AC + M[*251*]

The 3rd Instruction : STA 251 – Fetch Cycle



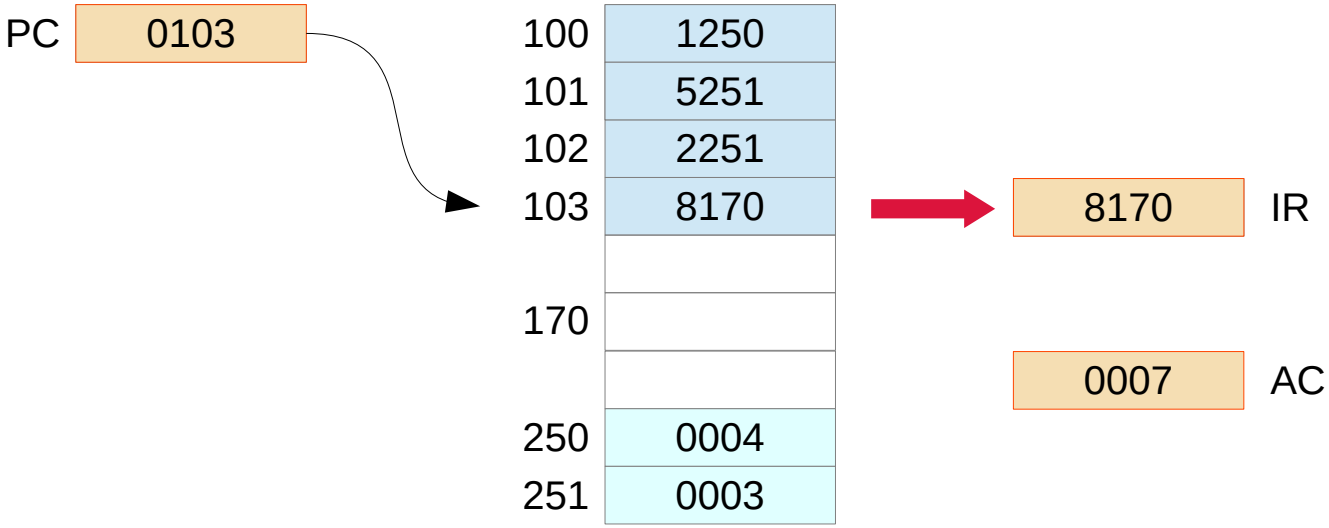
STA ***251*** ; M[*251*] ← AC

The 3rd Instruction : STA 251 – Execution Cycle



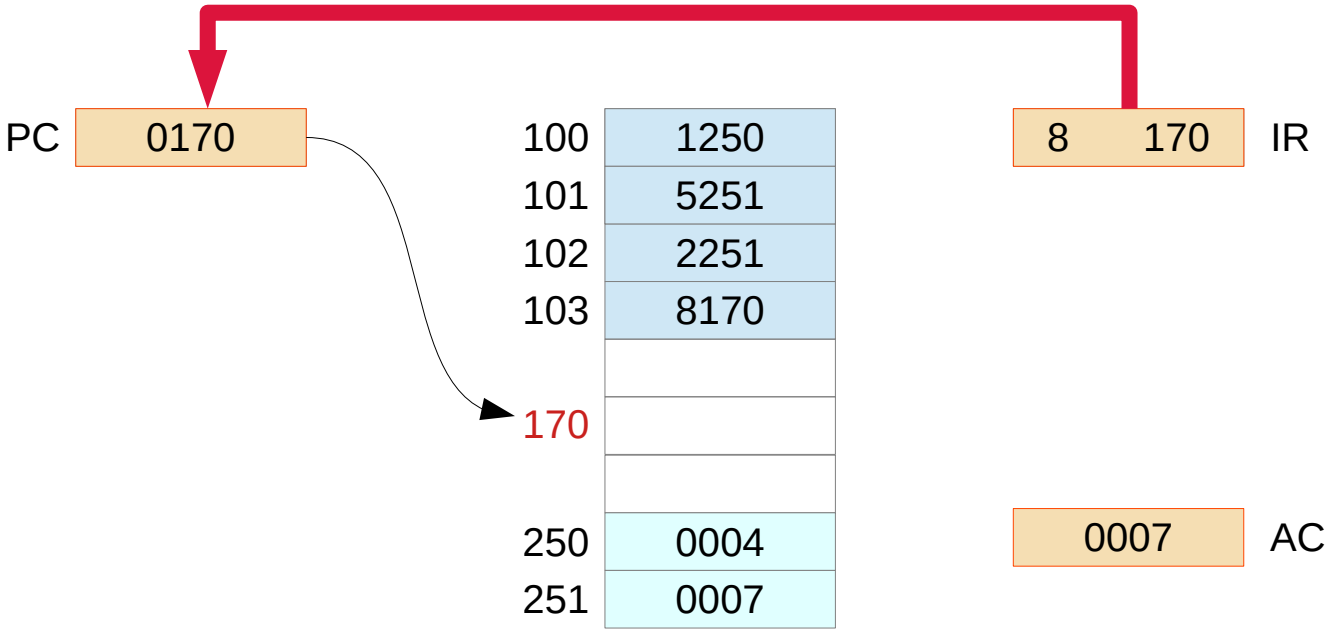
STA ***251*** ; M[*251*] ← AC

The 4th Instruction: JUMP 170 – Fetch Cycle



JUMP ***170*** ; PC ← *170*

The 4th Instruction: JUMP 170 – Execution Cycle



JUMP ***170*** ; PC ← *170*

References

[1] http://en.wikipedia.org/
[2] https://en.wikiversity.org/wiki/The_necessities_in_SOC_Design
[3] https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design
[4] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design
[5] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Architecture
[6] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Organization
[7] https://en.wikiversity.org/wiki/Understanding_Embedded_Software
[8] Digital Systems, Hill, Peterson, 1987
[9] http://en.wikipedia.org/
[10] http://www.ele.uri.edu/Courses/ele306/f01/Tinydoc.pdf