CPE 323
Intro to Embedded Computer Systems
MSP430 Instruction Set Architecture

Aleksandar Milenkovic

milenka@uah.edu

Admin

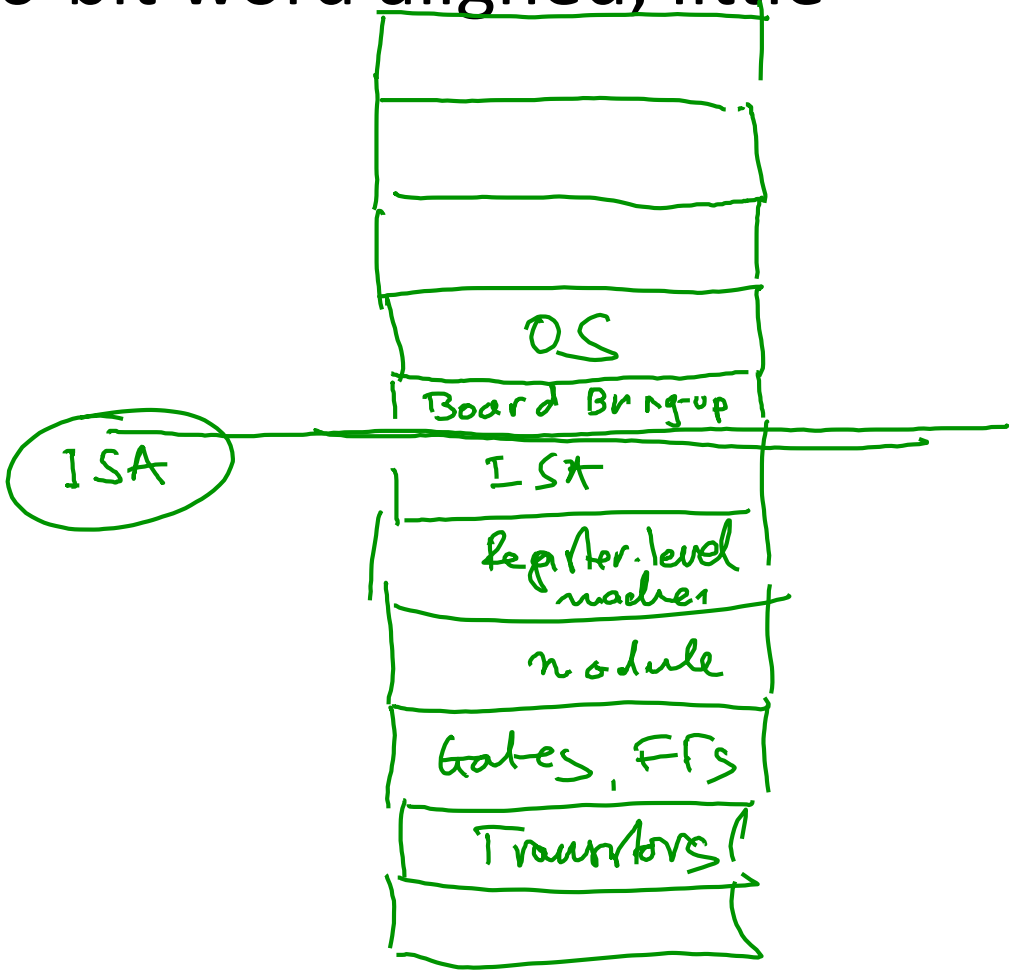
→ Quiz 1b

→ HW. 1

→ Quiz 2

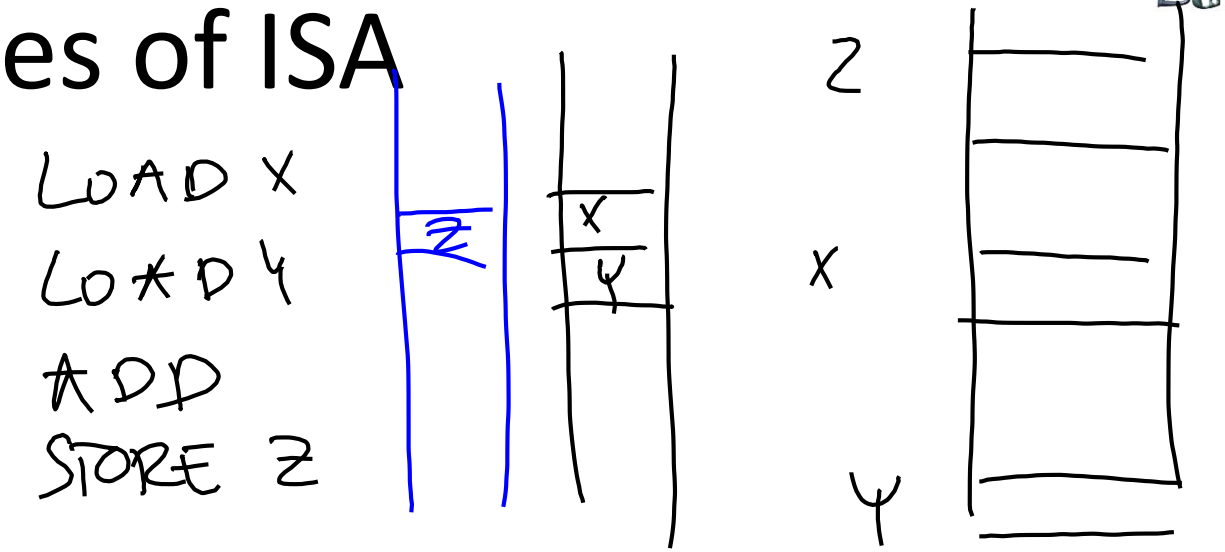
MSP430 Instruction Set Architecture

- 1. Types of ISA (16, 16-bit GPRs, R0=PC, R1=SP, R2=SR, R3=CG)
- 2. Memory View (byte addressable, 16-bit word aligned, little-endian)
- 3. Data Types (8-bit, 16-bit numbers)
- 4. Instruction Set
- 5. Addressing Modes
- 6. Instruction Encoding
- 7. Exceptions

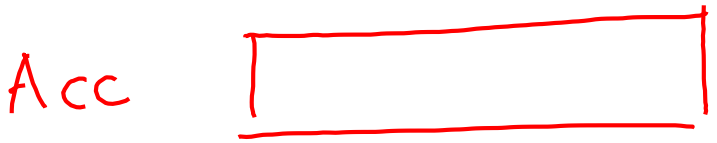


Types of ISA

- Stack $Z = X + Y$



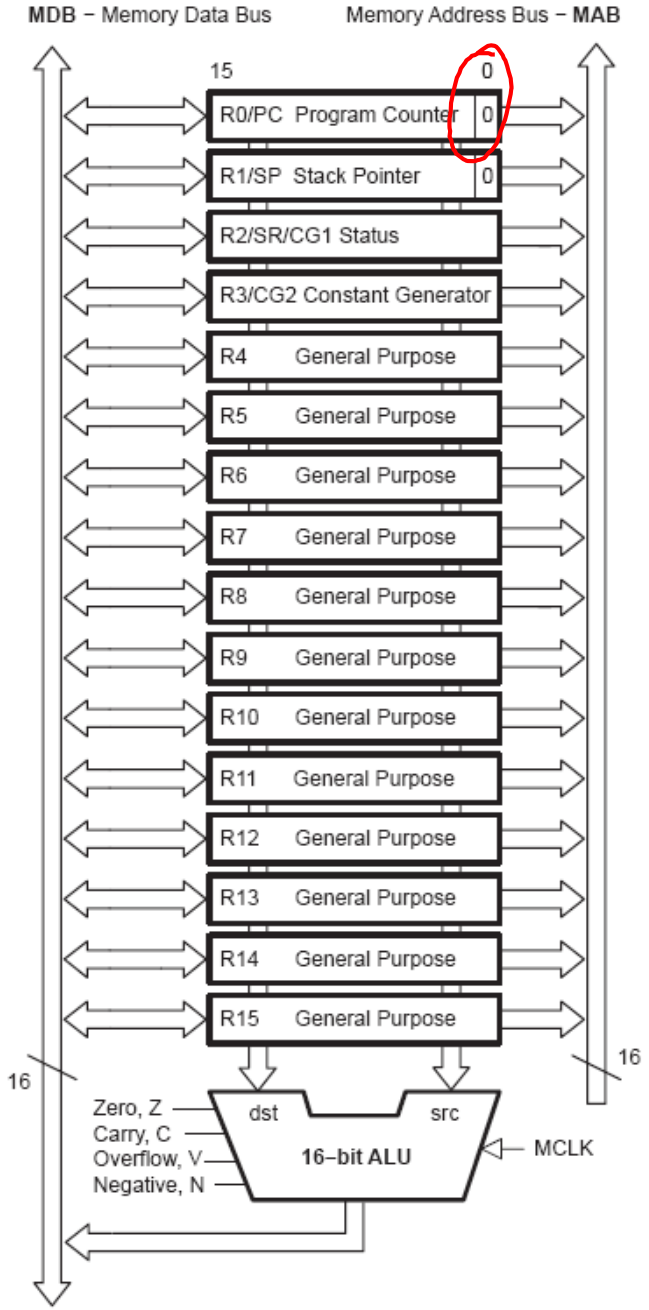
- Accumulator



LOAD X ; Acc ← M[X]
ADD Y ; Acc ← Acc + M[Y]
STORE Z ; M[Z] ← Acc

- Register/memory

MOV X, R5
MOV Y, R6
ADD R5, R6 ; R6 ← R6 + R5
STORE R6, Z



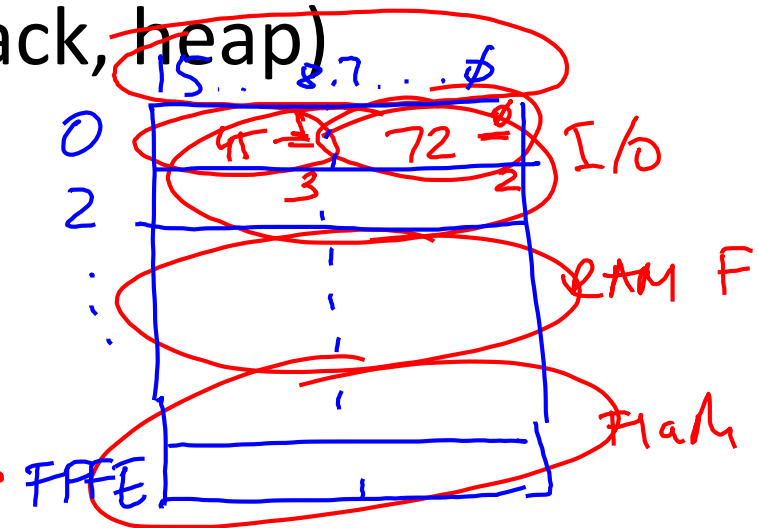
MSP430 Registers

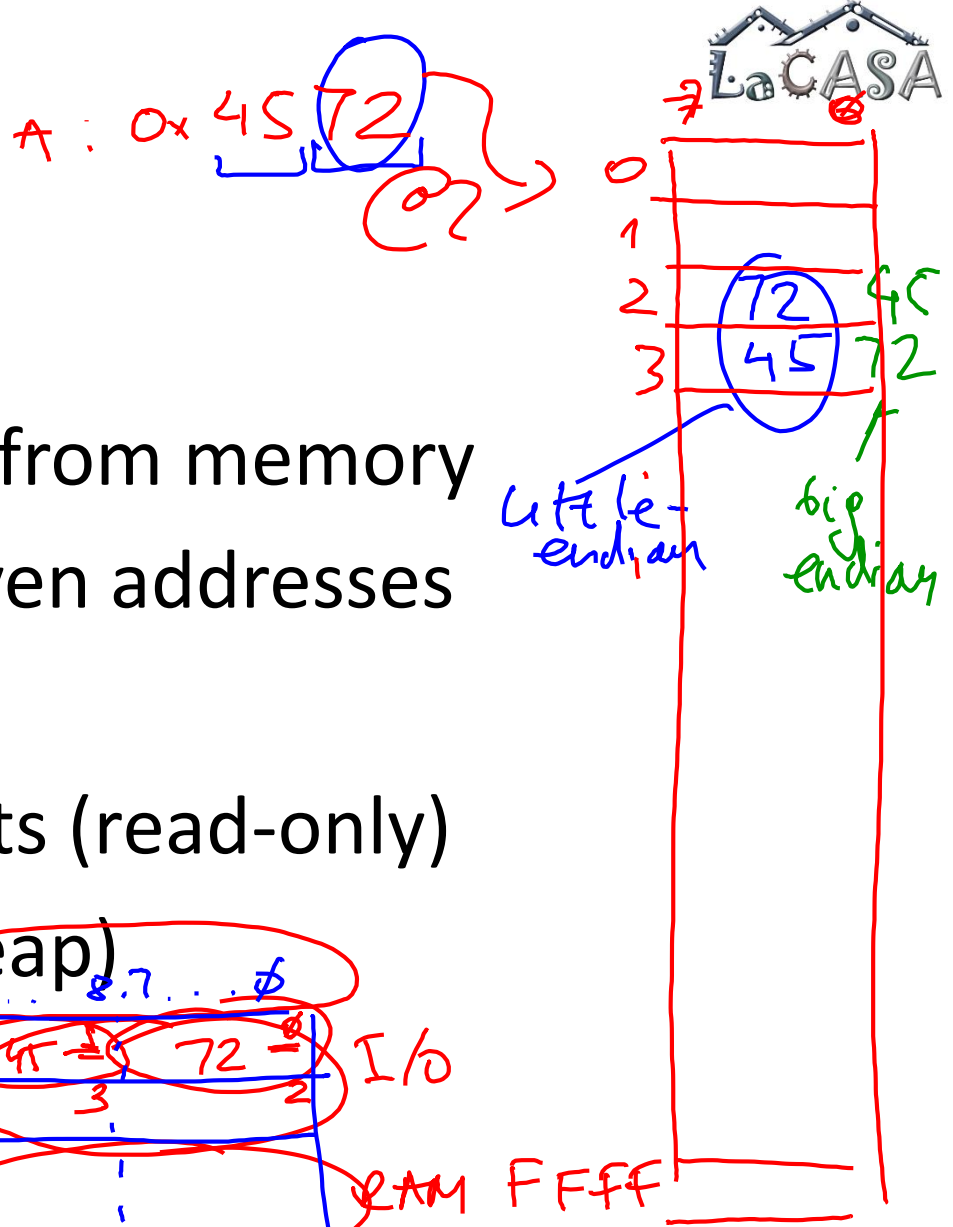
16, 16-bit registers R0 - R15

- R0 – Program Counter (R0, PC)
- R1 – Stack Pointer (R1, SP)
- R2 – Status Register (R2, SR)
- R3 – Constant Generator

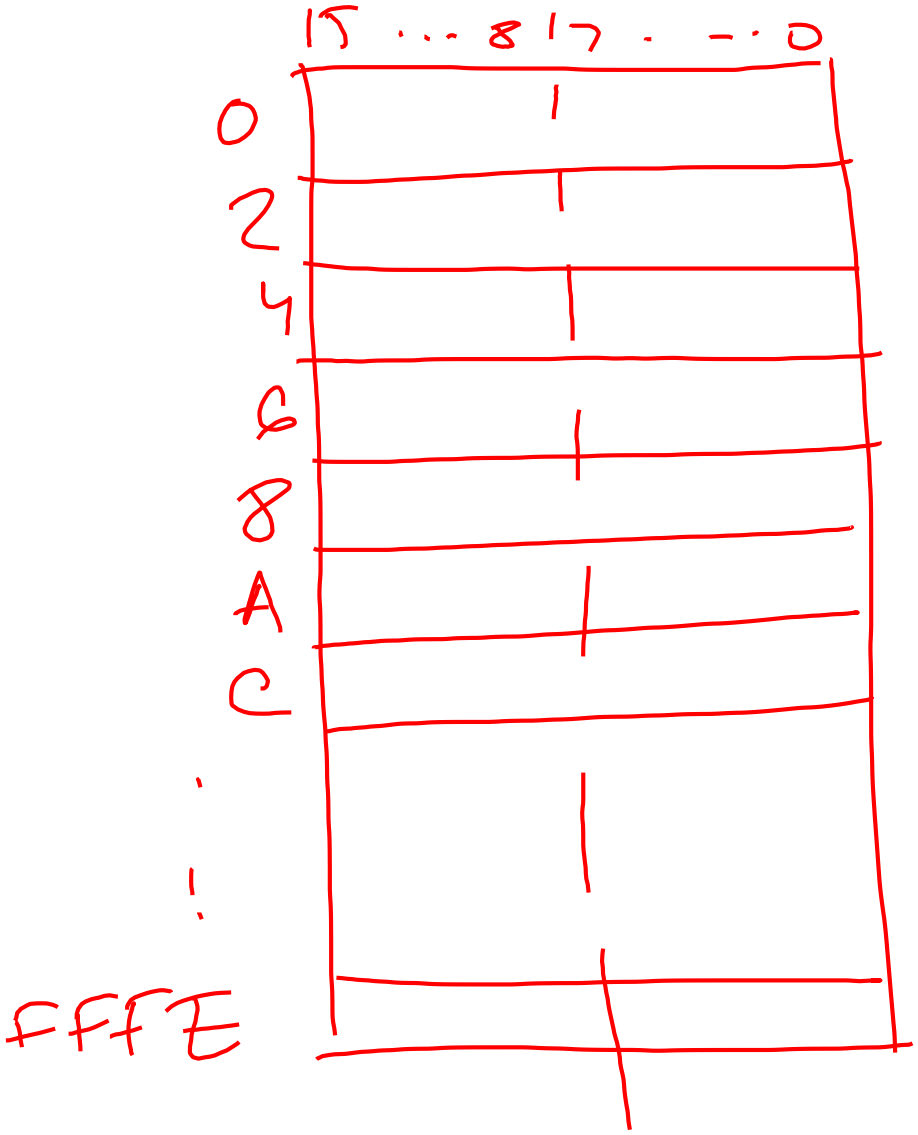
Memory

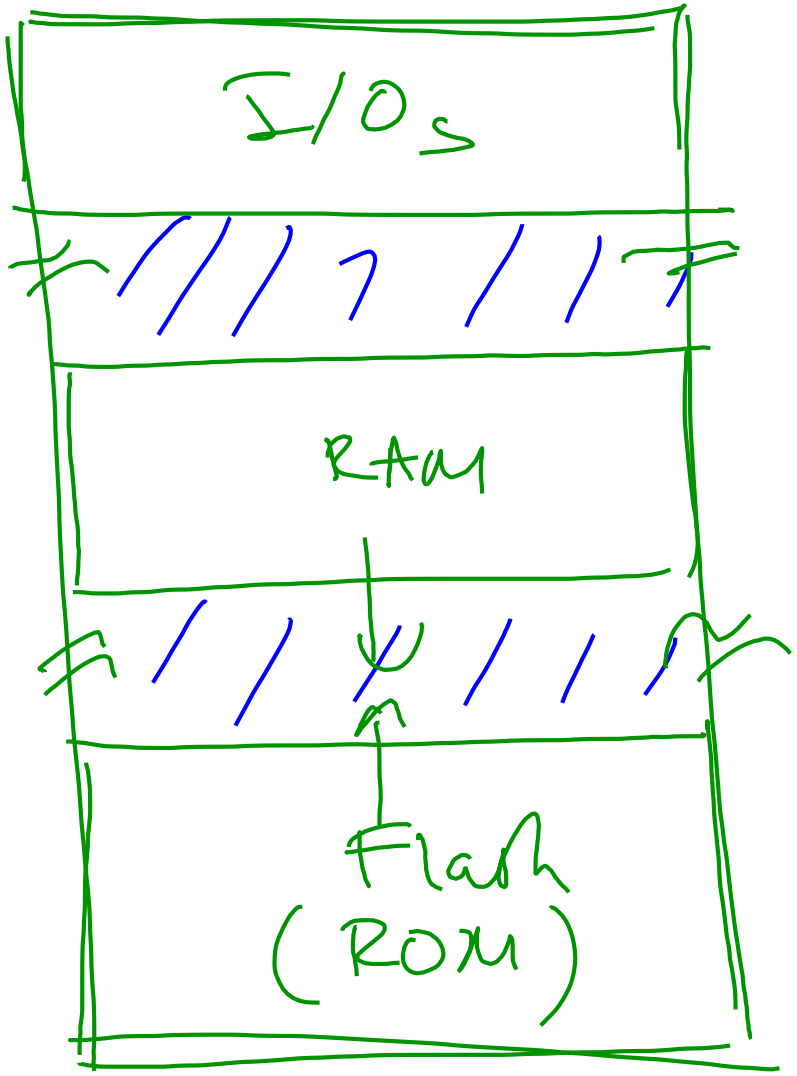
- Address space 2^16 bytes
- Byte addressable, can read 16-bit words from memory
- Words are aligned in memory: start at even addresses
- Little-endian placement policy
- Flash (ROM): Contains code and constants (read-only)
- RAM: Random Access Memory (stack, heap)
- I/O address space



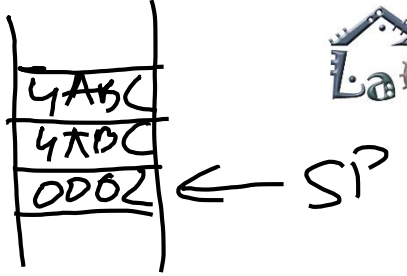


Memory

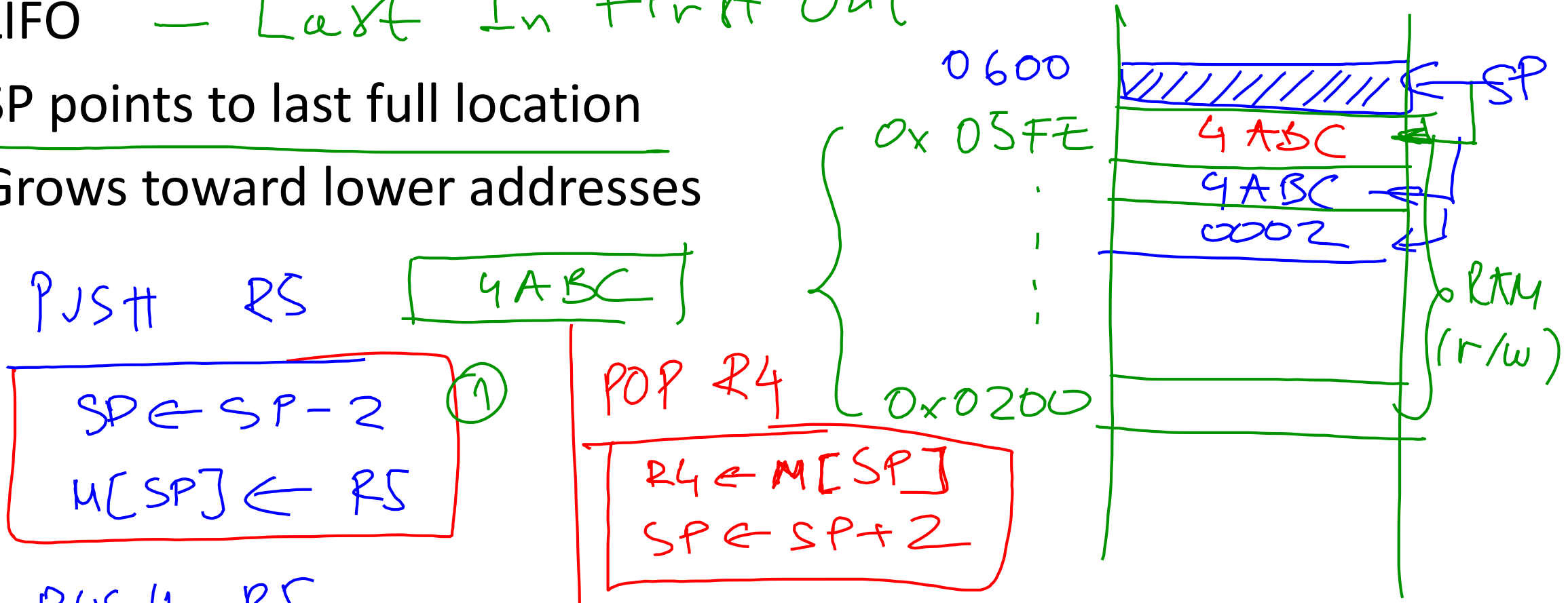




MSP430 Stack



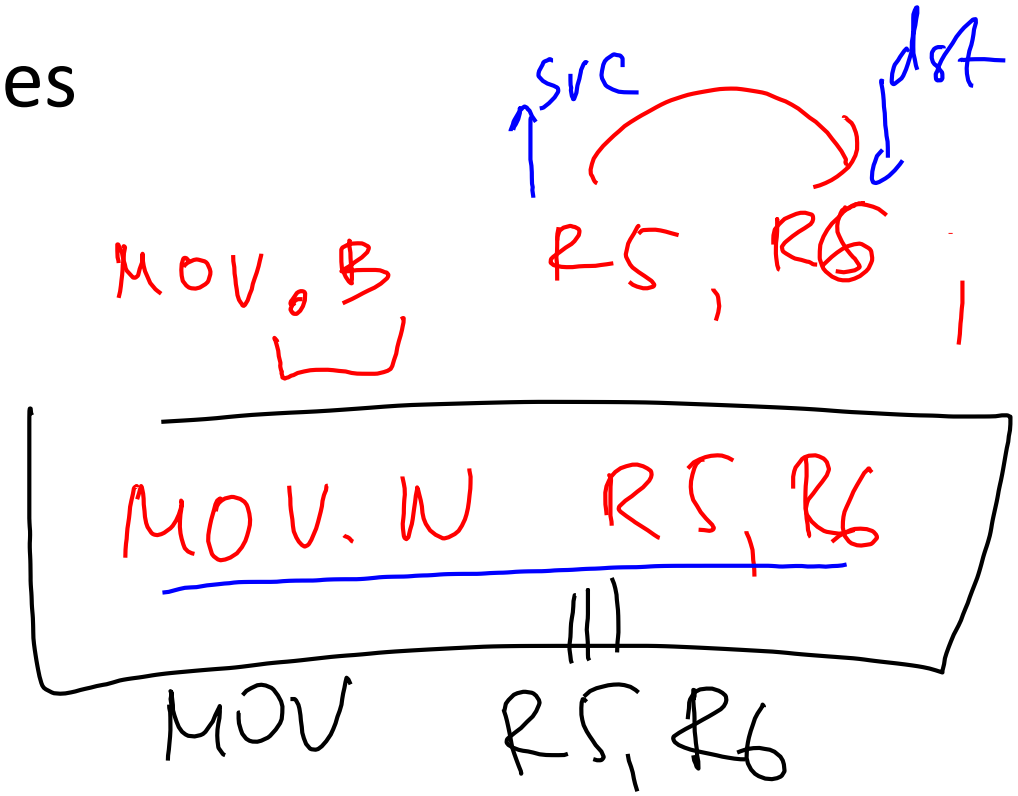
- LIFO — Last In First Out
- SP points to last full location
- Grows toward lower addresses

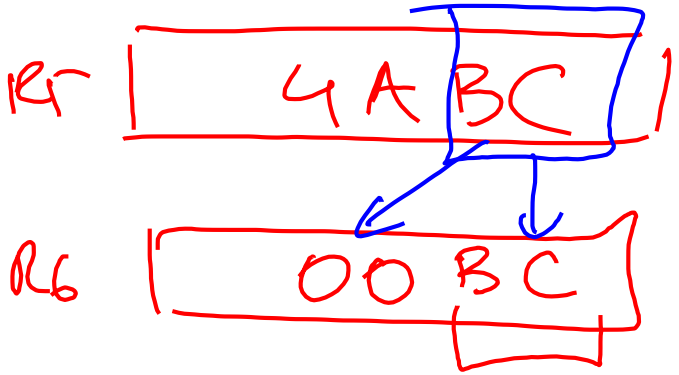


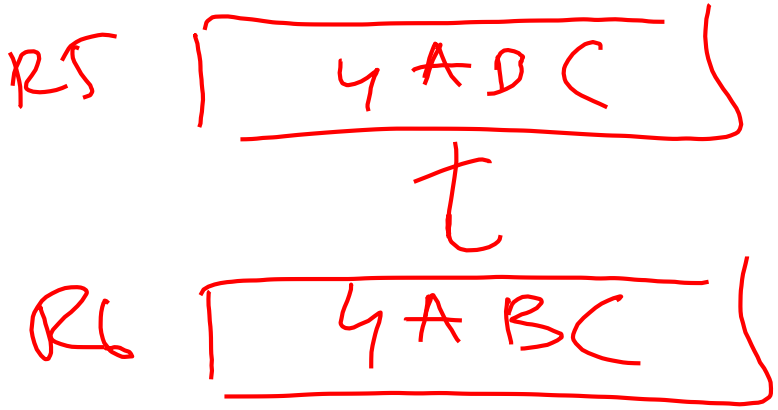
PUSH R5
PUSH #2

Data Types

- Instructions operating on bytes: suffix .b
- Instructions operating on words: suffix .w
- Examples

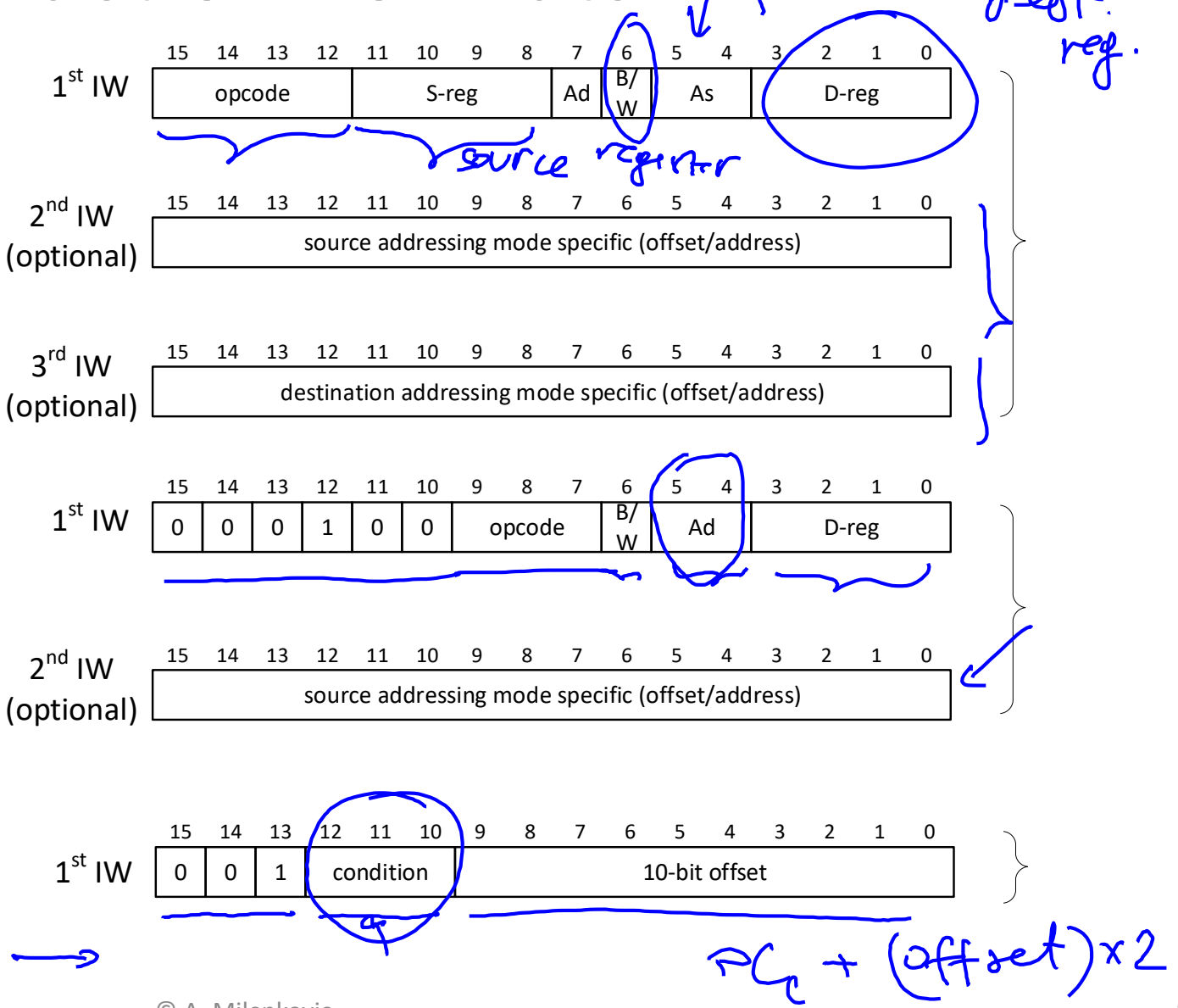






Instruction Formats

- Double-operand
- Single-operand
- Jumps

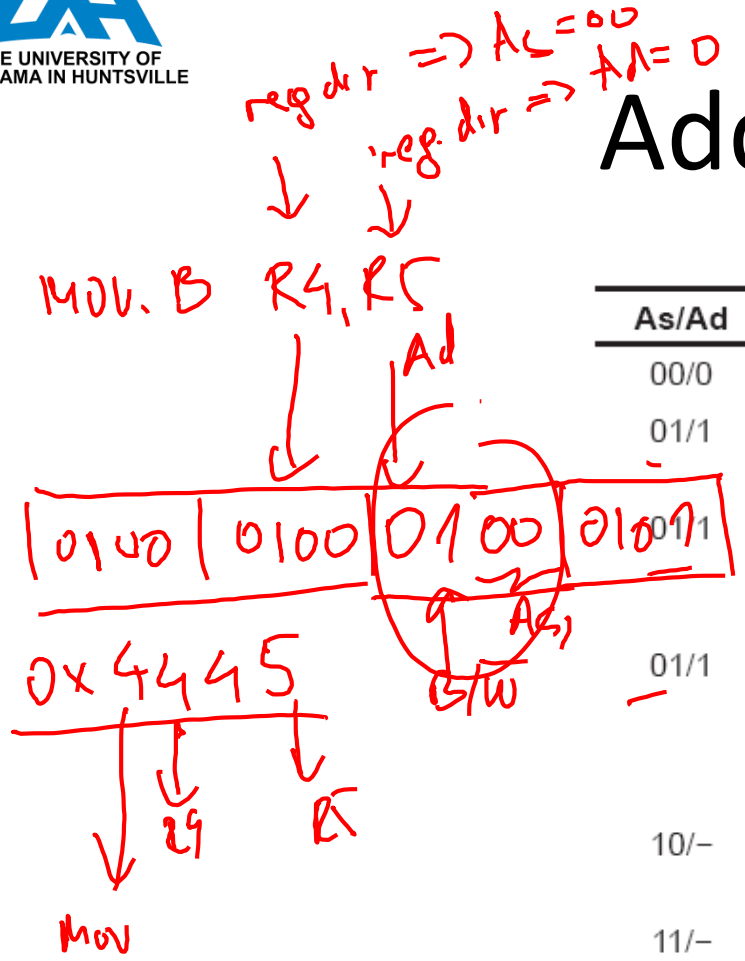


Addressing Modes

Name	Source Operand	Destination Operand
Register MOV.B R4, R6	✓	✓
Indexed MOV.B 400(R4), R6	✓	✓
Symbolic MOV.B MYS, R6	✓	✓
Absolute MOV.B &MYS, R6	✓	✓
Register Indirect MOV.B @R5, R6	✓	X
Autoincrement (Register indirect with autoincrement) MOV.B @R5+, R6	✓	X
Immediate MOV.B #45, R6	✓	X

$EA_S = 400 + R4$

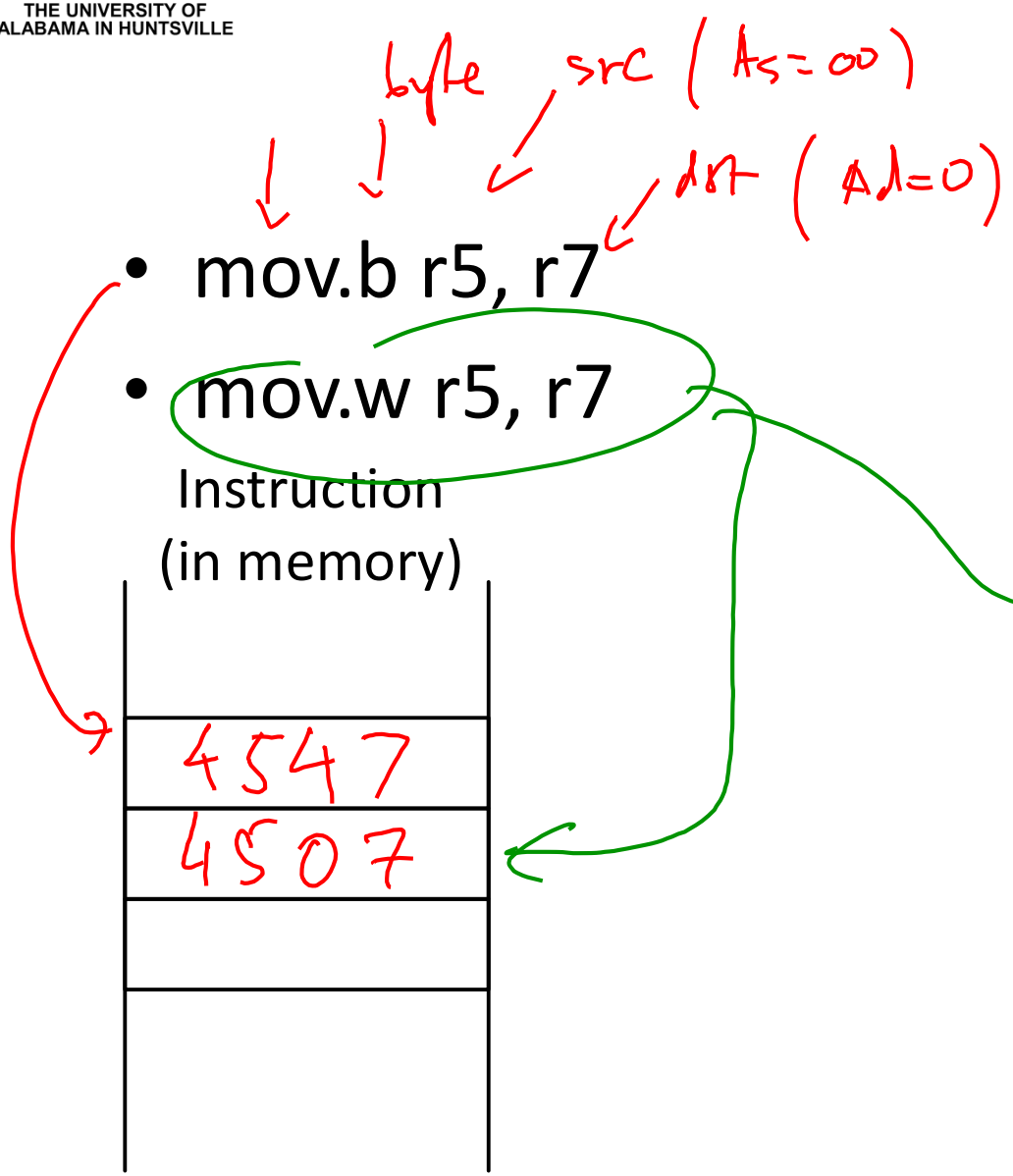
Address Specifiers (As, Ad)

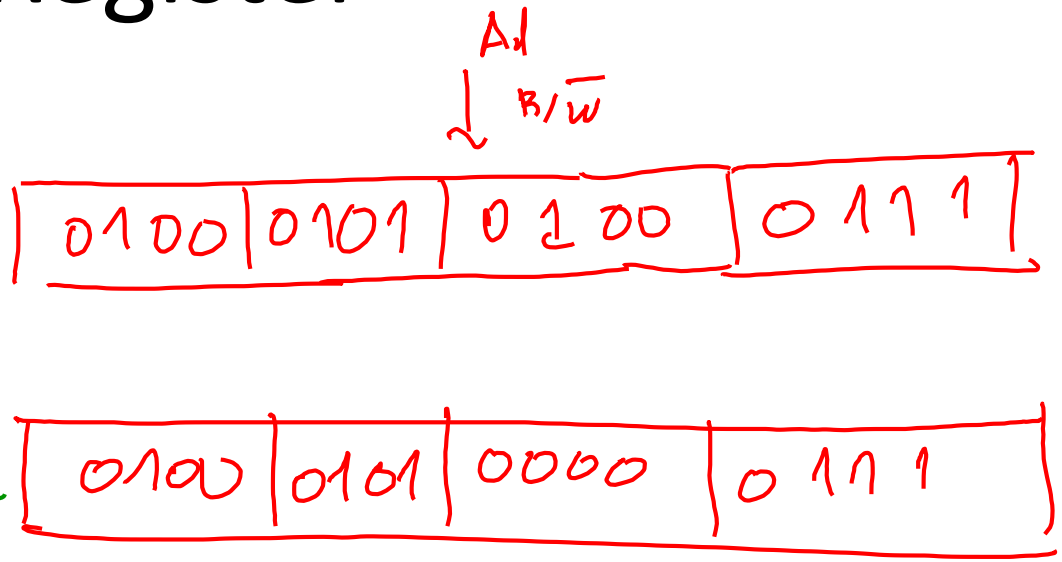


As/Ad	Addressing Mode	Syntax	Description
00/0	Register mode	Rn	Register contents are operand
01/1	Indexed mode	X(Rn)	(Rn + X) points to the operand. X is stored in the next word.
01/1	Symbolic mode	ADDR	(PC + X) points to the operand. X is stored in the next word. Indexed mode X(PC) is used.
01/1	Absolute mode	&ADDR	The word following the instruction contains the absolute address. X is stored in the next word. Indexed mode X(SR) is used.
10/–	Indirect register mode	@Rn	Rn is used as a pointer to the operand.
11/–	Indirect autoincrement	@Rn+	Rn is used as a pointer to the operand. Rn is incremented afterwards by 1 for .B instructions and by 2 for .W instructions.
11/–	Immediate mode	#N	The word following the instruction contains the immediate constant N. Indirect autoincrement mode @PC+ is used.

Register

- mov.b r5, r7
- mov.w r5, r7





Indexed

- mov.w 0x100(r4), 0x200(r5) ; M[0x200+r5] ← M[0x100+r4]

Indexed As=01

Indexed Ad=1

Instruction (in memory)

4495
0100
0200

$EA_S = 0x100 + r4$

$EA_d = 0x200 + r5$

1st word: 0100 | 0100 | 1001 | 0101

4495

Memory

A23F
A23F

Symbolic

symbolic symbolic

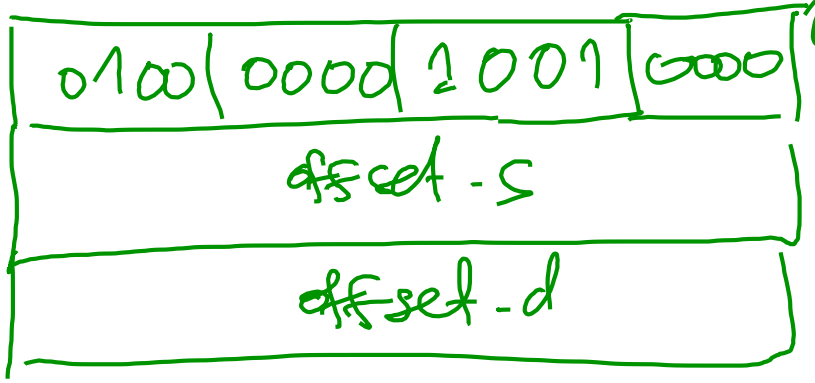
- mov.w A, B ; M[B] ← M[A]

MOV.W 0xF000, 0x0200

Memory

Ad

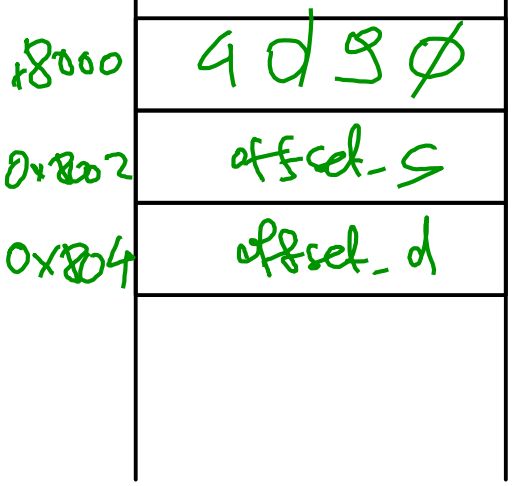
1st word:

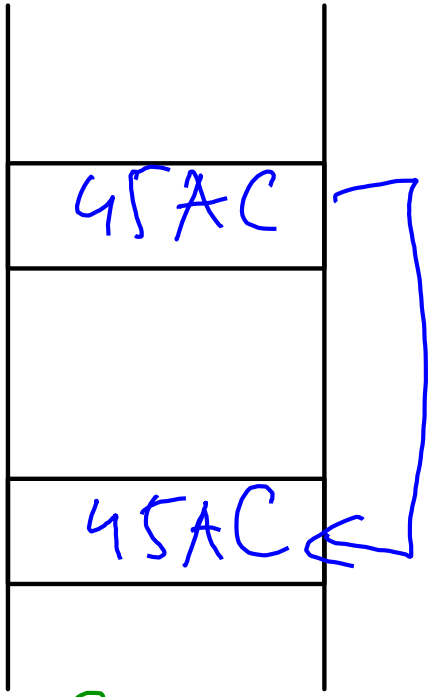


0x F000 ≡ A

0x0200 ≡ B

Instruction (in memory)



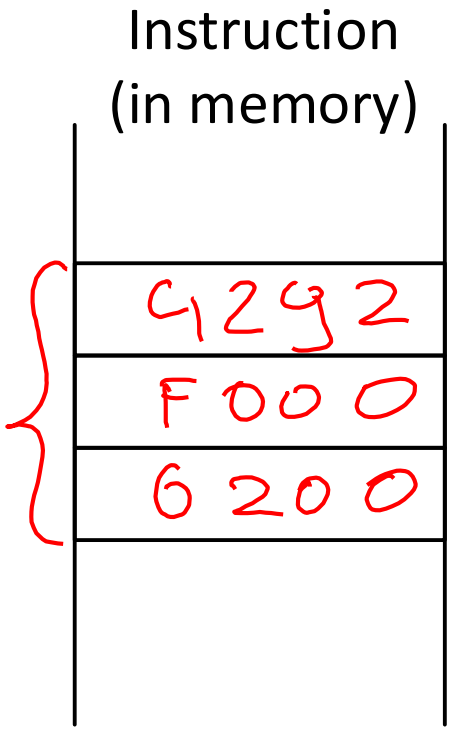


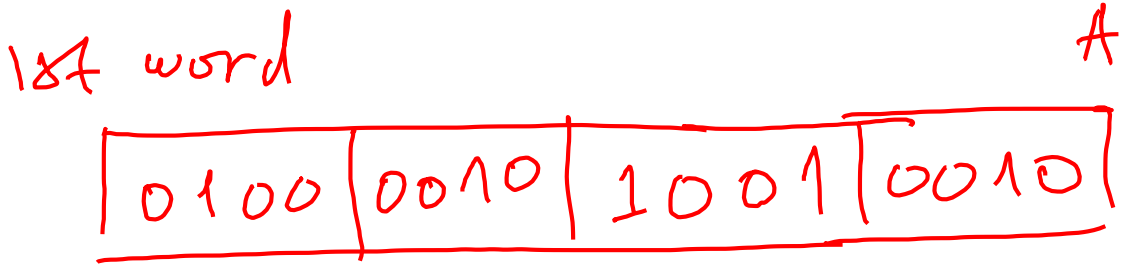
EA_S = 0xF000 = 0x8002 + offset_S

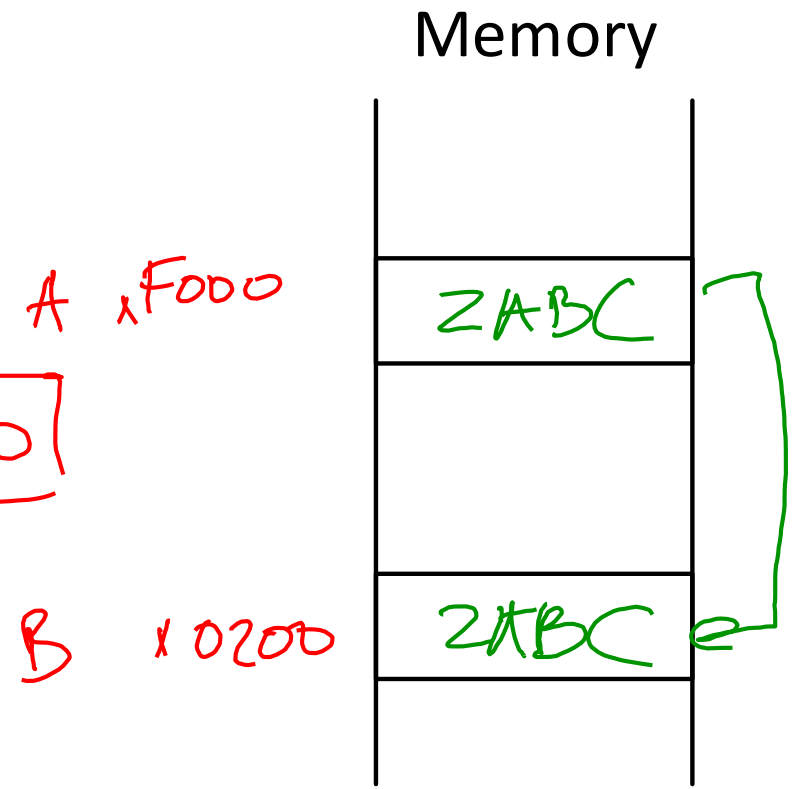
EA_d = 0x0200 = 0x8004 + offset_d

Absolute

- mov.w &A, &B ; M[B] ← M[A]

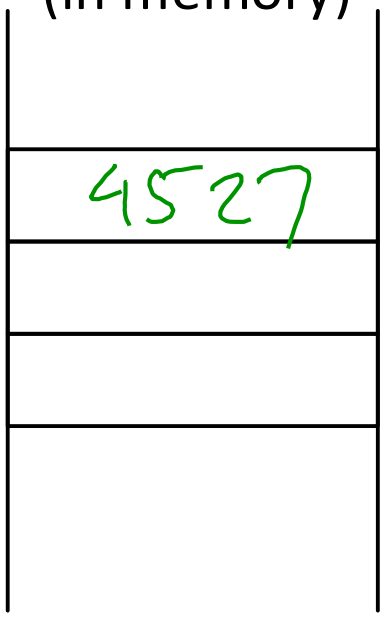


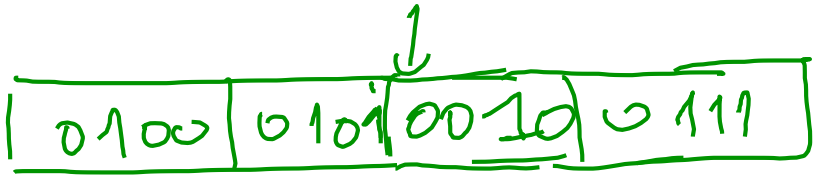


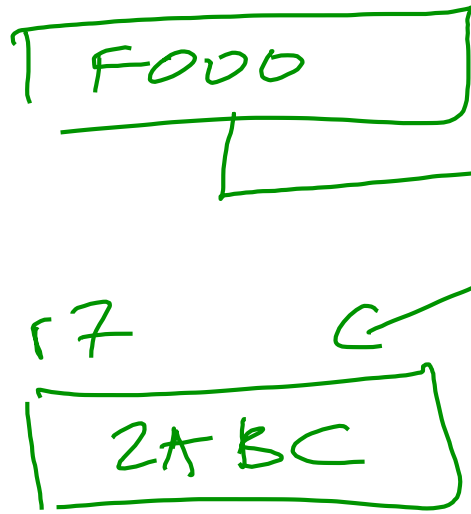


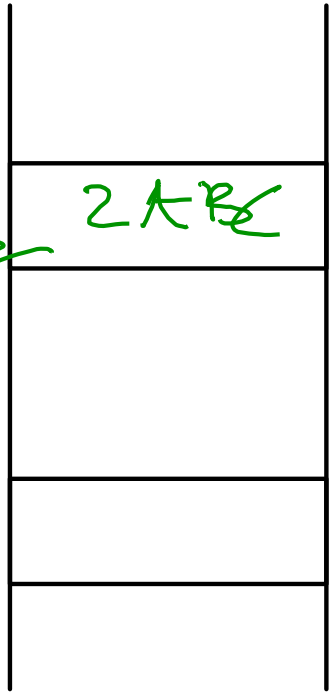
Register Indirect

- mov.w @r5, r7



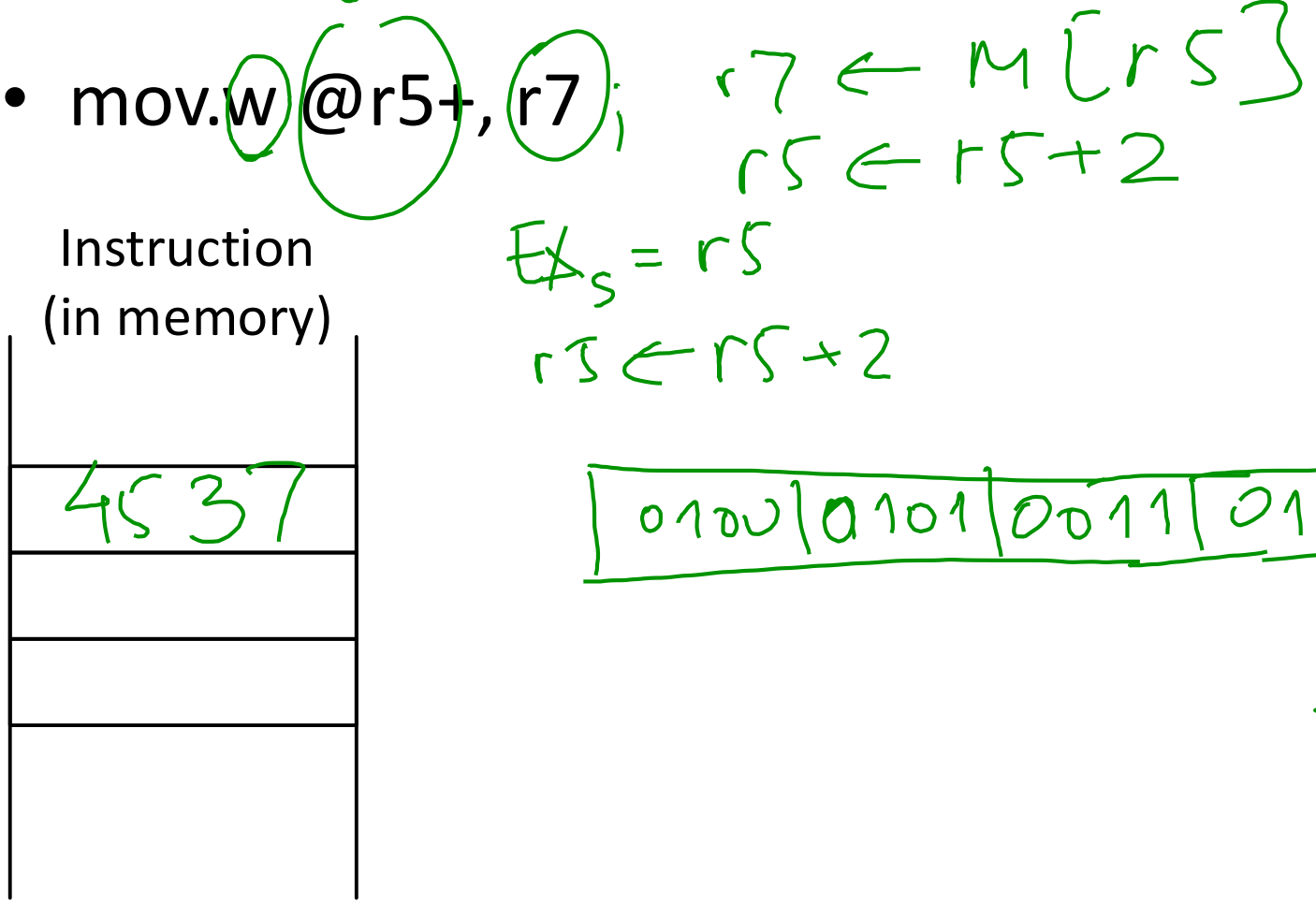






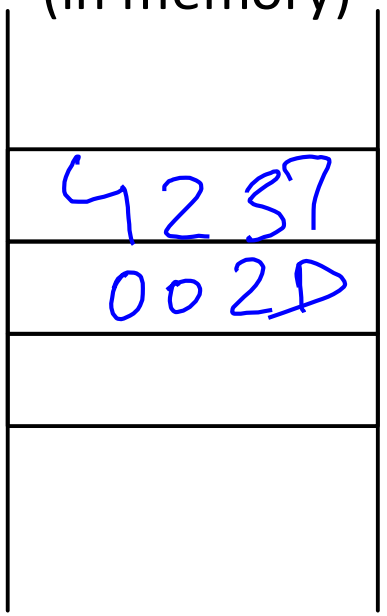
Autoincrement

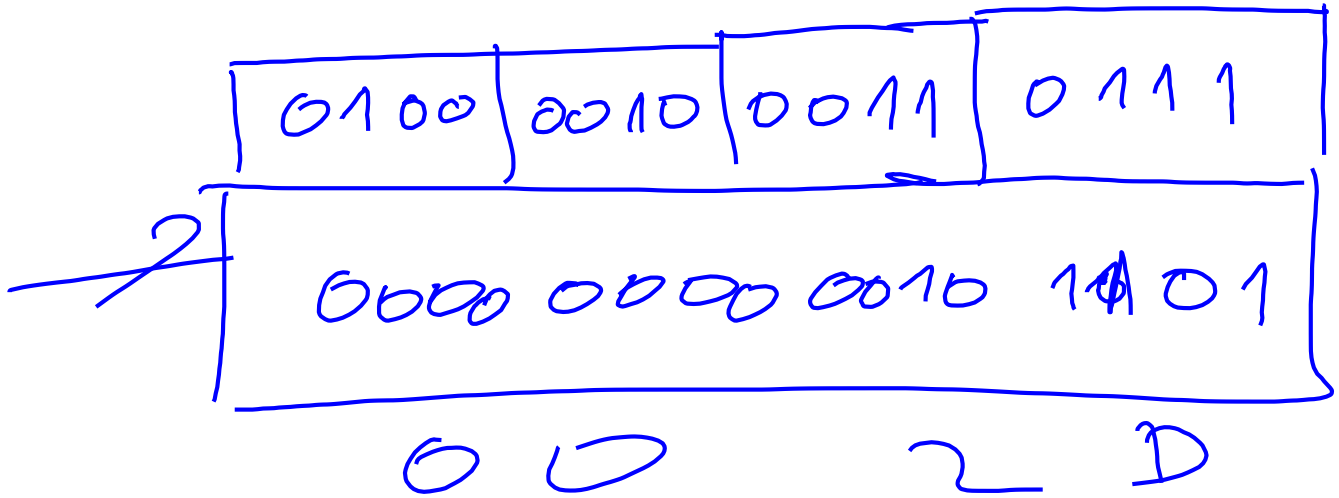
- mov.w @r5+, r7

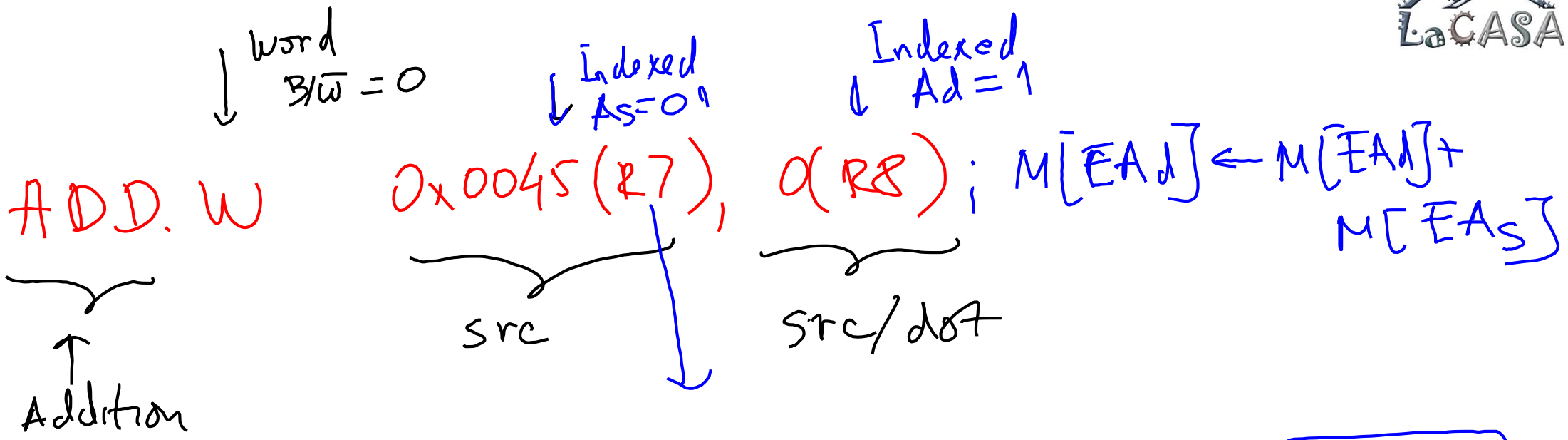


Immediate

- mov.w #45, r7







1st word:

xxxx	0111	1001	1000

?798
0045
0000

$EA_S = R7 + 0x45$

$EA_d = R8 + 0$

Double Operand Instruction

MOV(.B) src, dst ; dst ← src [does not affect flags]

ADD(.B) src, dst ; dst ← src + dst [binary]

ADDC(.B) src, dst ; Dst ← src + dst + C

SUB(.B) src, dst ; dst ← dst + $\overline{src}$ + 1

SUBC(.B) src, dst ; dst ← dst + $\overline{src}$ + C

CMP(.B) src, dst ; dst - src ≡ dst + $\overline{src}$ + 1

DADD(.B) src, dst ; dst ← dst + src + C (decimal)

BIT(.B) src, dst ; src and dst

BIC, BIS, XOR, AND bit clear