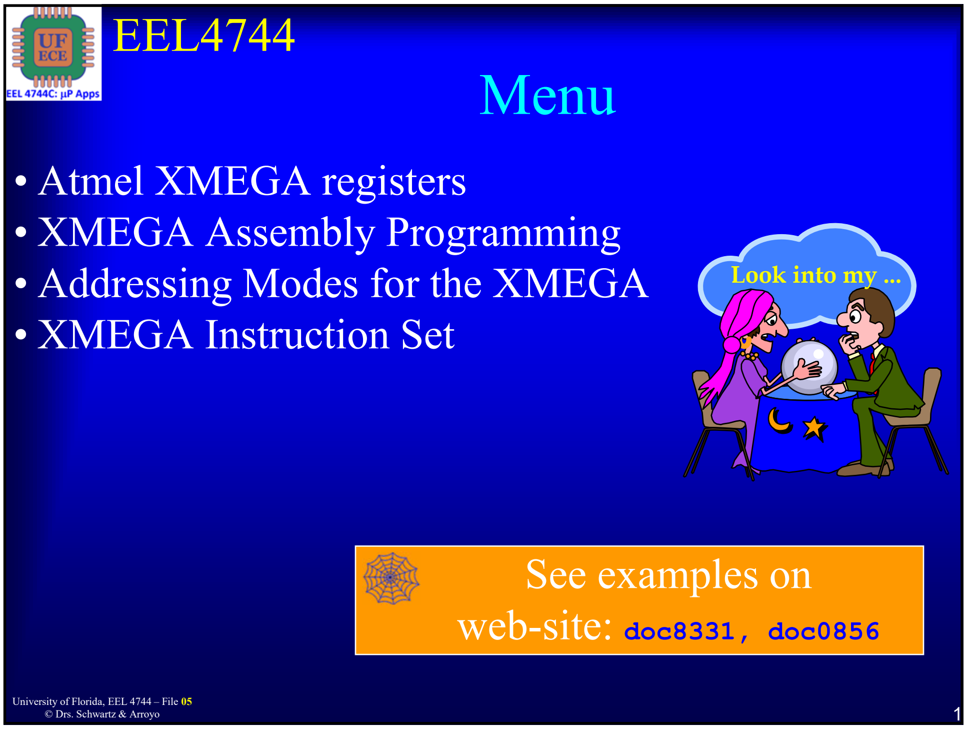


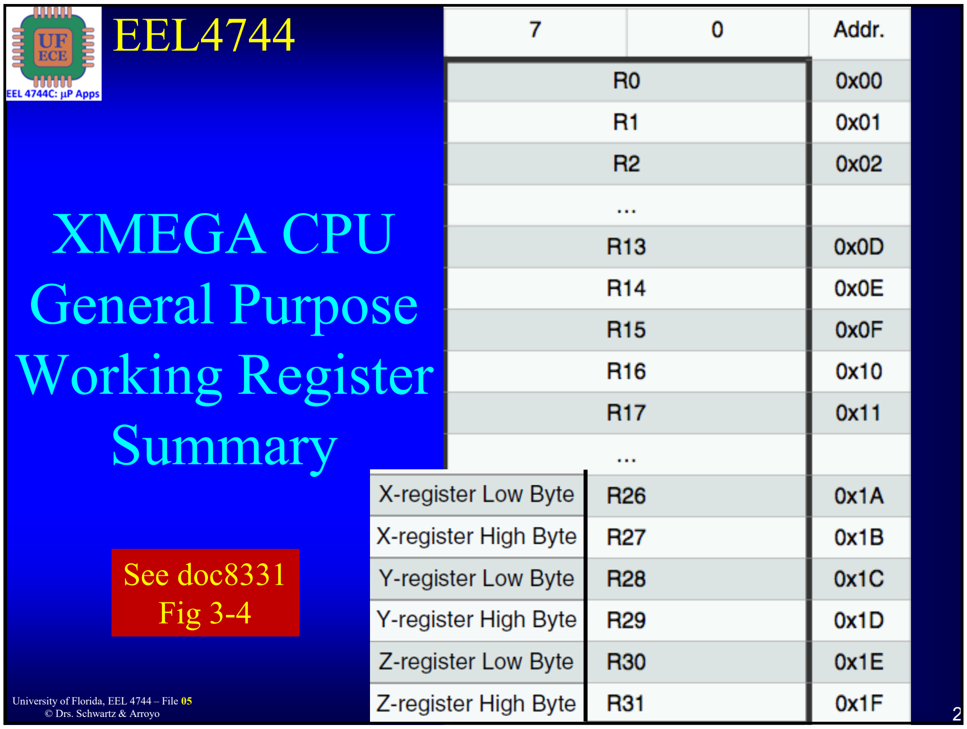
EEL4744

Menu

- Atmel XMEGA registers
- XMEGA Assembly Programming
- Addressing Modes for the XMEGA
- XMEGA Instruction Set

Look into my ...

See examples on web-site: `doc8331, doc0856`

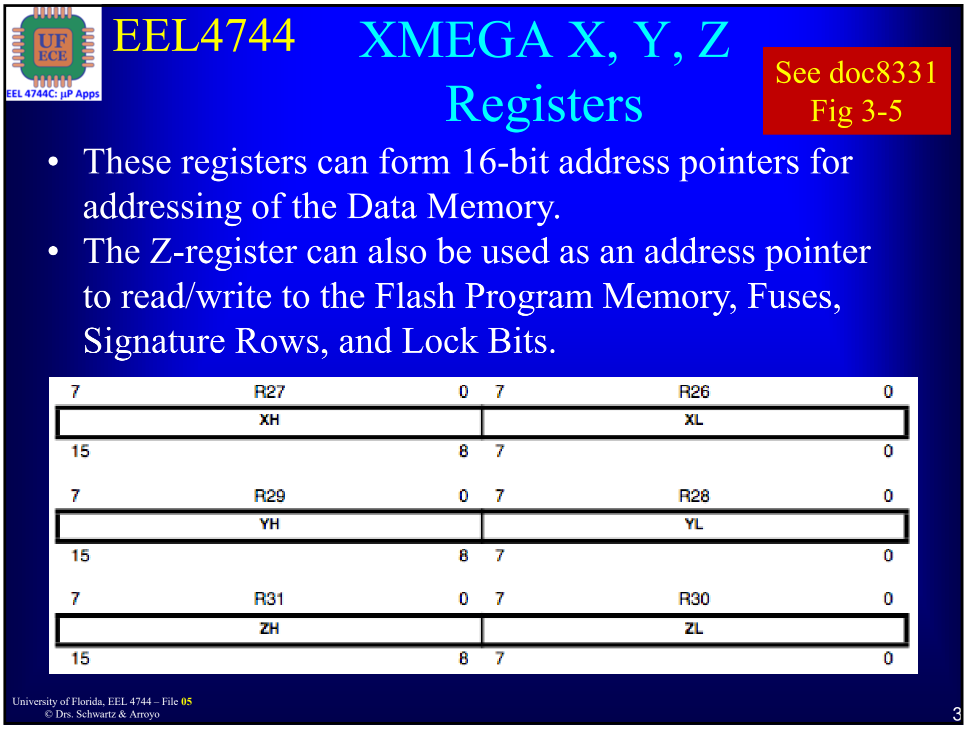


EEL4744

XMEGA CPU General Purpose Working Register Summary

See doc8331 Fig 3-4

	7	0	Addr.
	R0		0x00
	R1		0x01
	R2		0x02
	...		
	R13		0x0D
	R14		0x0E
	R15		0x0F
	R16		0x10
	R17		0x11
	...		
X-register Low Byte	R26		0x1A
X-register High Byte	R27		0x1B
Y-register Low Byte	R28		0x1C
Y-register High Byte	R29		0x1D
Z-register Low Byte	R30		0x1E
Z-register High Byte	R31		0x1F



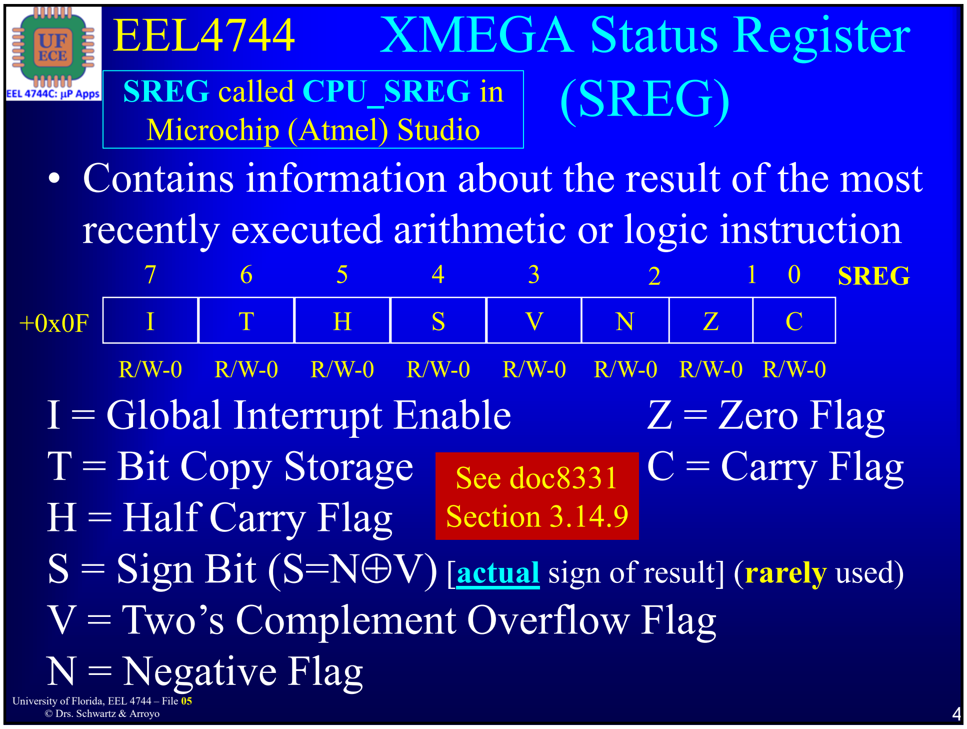
EEL4744 XMEGA X, Y, Z Registers

See doc8331 Fig 3-5

- These registers can form 16-bit address pointers for addressing of the Data Memory.
- The Z-register can also be used as an address pointer to read/write to the Flash Program Memory, Fuses, Signature Rows, and Lock Bits.

7	R27	0	7	R26	0
	XH			XL	
15		8	7		0
7	R29	0	7	R28	0
	YH			YL	
15		8	7		0
7	R31	0	7	R30	0
	ZH			ZL	
15		8	7		0

3



EEL4744 XMEGA Status Register (SREG)

SREG called CPU_SREG in Microchip (Atmel) Studio

- Contains information about the result of the most recently executed arithmetic or logic instruction

	7	6	5	4	3	2	1	0	SREG
+0x0F	I	T	H	S	V	N	Z	C	
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

I = Global Interrupt Enable

T = Bit Copy Storage

H = Half Carry Flag

S = Sign Bit (S=N⊕V) [**actual** sign of result] (**rarely** used)

V = Two's Complement Overflow Flag

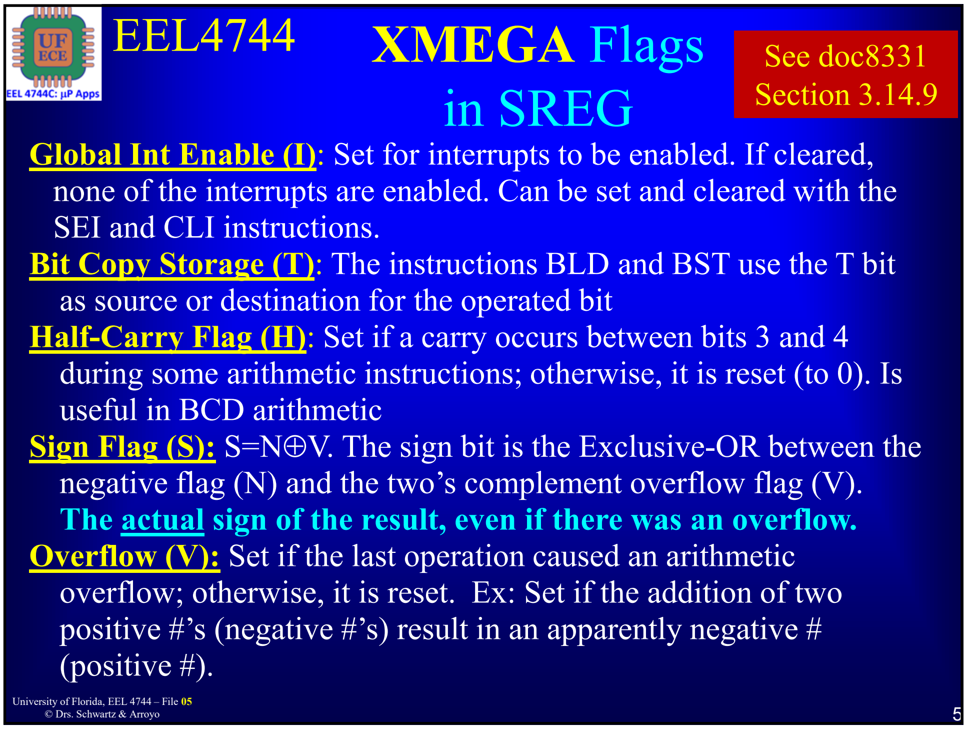
N = Negative Flag

Z = Zero Flag

C = Carry Flag

See doc8331 Section 3.14.9

4



EEL4744

XMEGA Flags in SREG

See doc8331
Section 3.14.9

Global Int Enable (I): Set for interrupts to be enabled. If cleared, none of the interrupts are enabled. Can be set and cleared with the SEI and CLI instructions.

Bit Copy Storage (T): The instructions BLD and BST use the T bit as source or destination for the operated bit

Half-Carry Flag (H): Set if a carry occurs between bits 3 and 4 during some arithmetic instructions; otherwise, it is reset (to 0). Is useful in BCD arithmetic

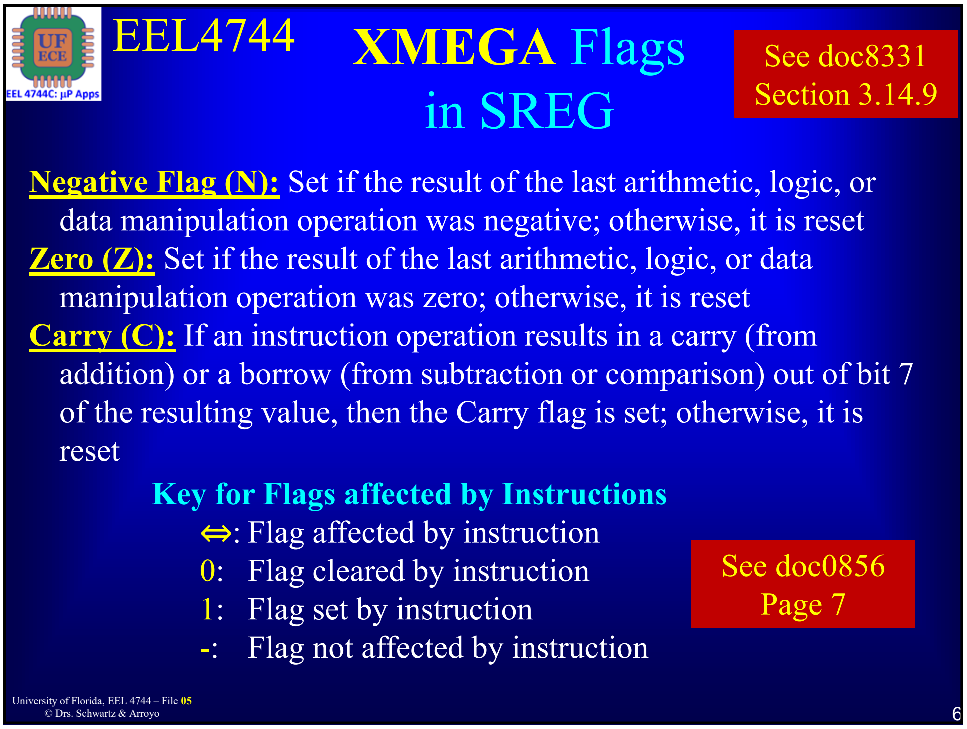
Sign Flag (S): S=N⊕V. The sign bit is the Exclusive-OR between the negative flag (N) and the two's complement overflow flag (V). **The actual sign of the result, even if there was an overflow.**

Overflow (V): Set if the last operation caused an arithmetic overflow; otherwise, it is reset. Ex: Set if the addition of two positive #'s (negative #'s) result in an apparently negative # (positive #).

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo

5

5



EEL4744

XMEGA Flags in SREG

See doc8331
Section 3.14.9

Negative Flag (N): Set if the result of the last arithmetic, logic, or data manipulation operation was negative; otherwise, it is reset

Zero (Z): Set if the result of the last arithmetic, logic, or data manipulation operation was zero; otherwise, it is reset

Carry (C): If an instruction operation results in a carry (from addition) or a borrow (from subtraction or comparison) out of bit 7 of the resulting value, then the Carry flag is set; otherwise, it is reset

Key for Flags affected by Instructions

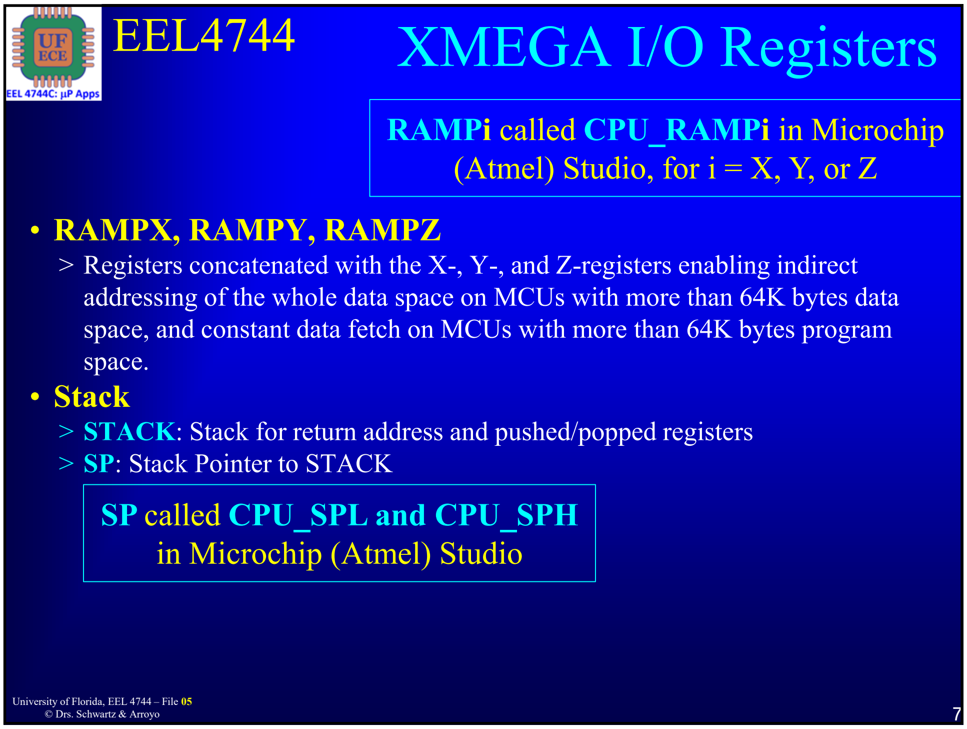
⇔:	Flag affected by instruction
0:	Flag cleared by instruction
1:	Flag set by instruction
-:	Flag not affected by instruction

See doc0856
Page 7

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo

6

6



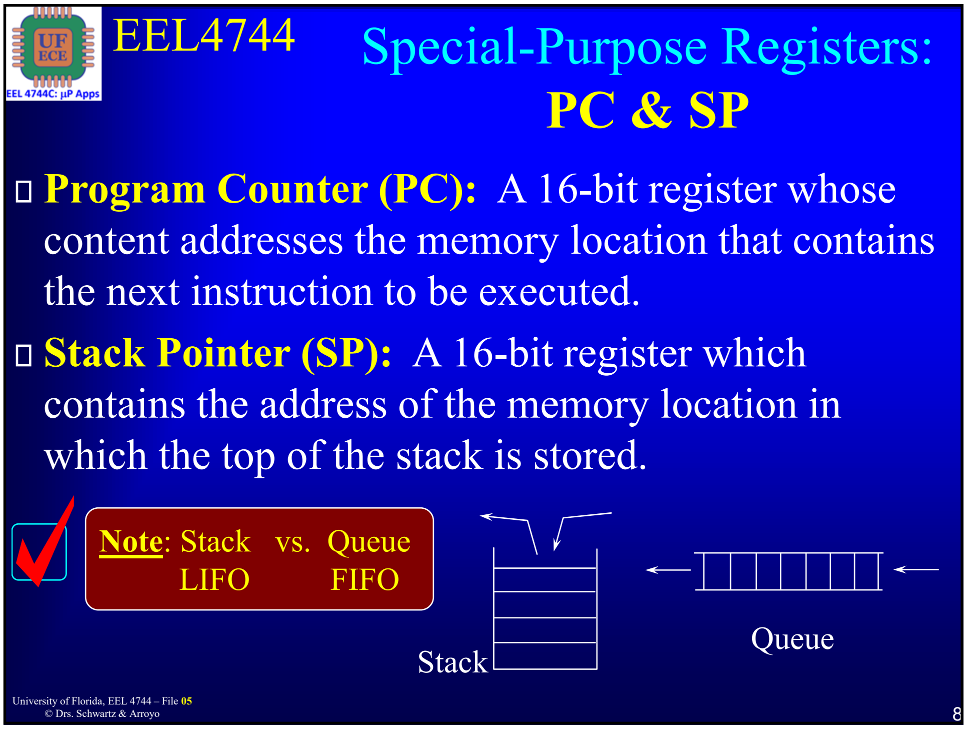
EEL4744

XMEGA I/O Registers

RAMPi called **CPU_RAMPi** in Microchip (Atmel) Studio, for i = X, Y, or Z

- **RAMPX, RAMPY, RAMPZ**
  > Registers concatenated with the X-, Y-, and Z-registers enabling indirect addressing of the whole data space on MCUs with more than 64K bytes data space, and constant data fetch on MCUs with more than 64K bytes program space.
- **Stack**
  > **STACK**: Stack for return address and pushed/popped registers
  > **SP**: Stack Pointer to STACK

SP called **CPU_SPL and CPU_SPH** in Microchip (Atmel) Studio



EEL4744

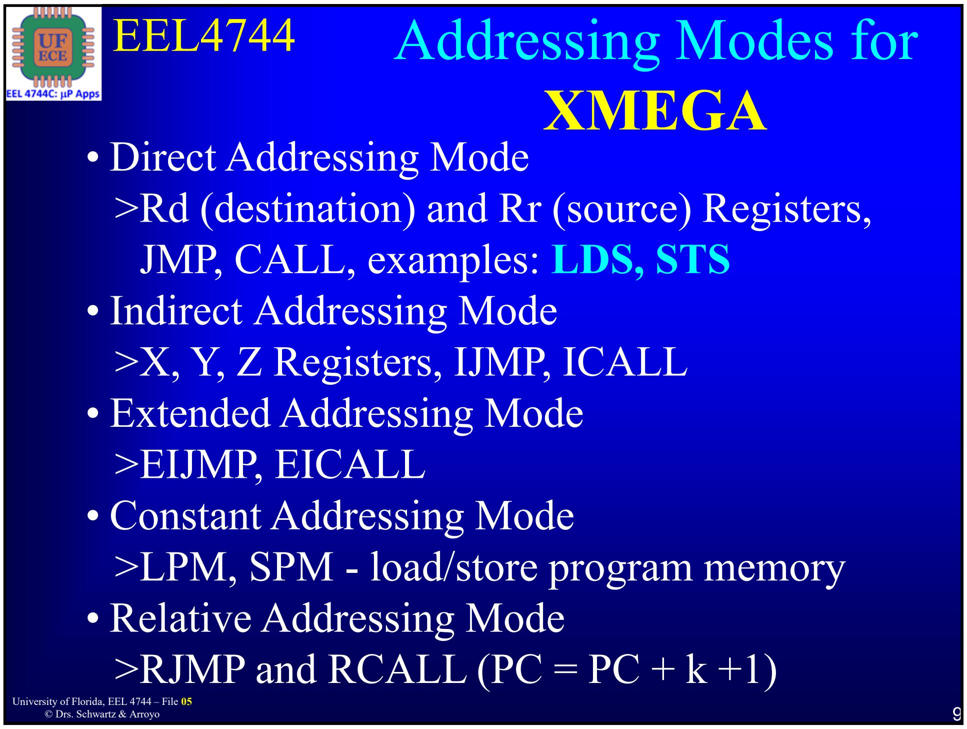
Special-Purpose Registers: PC & SP

- **Program Counter (PC):** A 16-bit register whose content addresses the memory location that contains the next instruction to be executed.
- **Stack Pointer (SP):** A 16-bit register which contains the address of the memory location in which the top of the stack is stored.

Note: Stack vs. Queue
LIFO FIFO

Stack

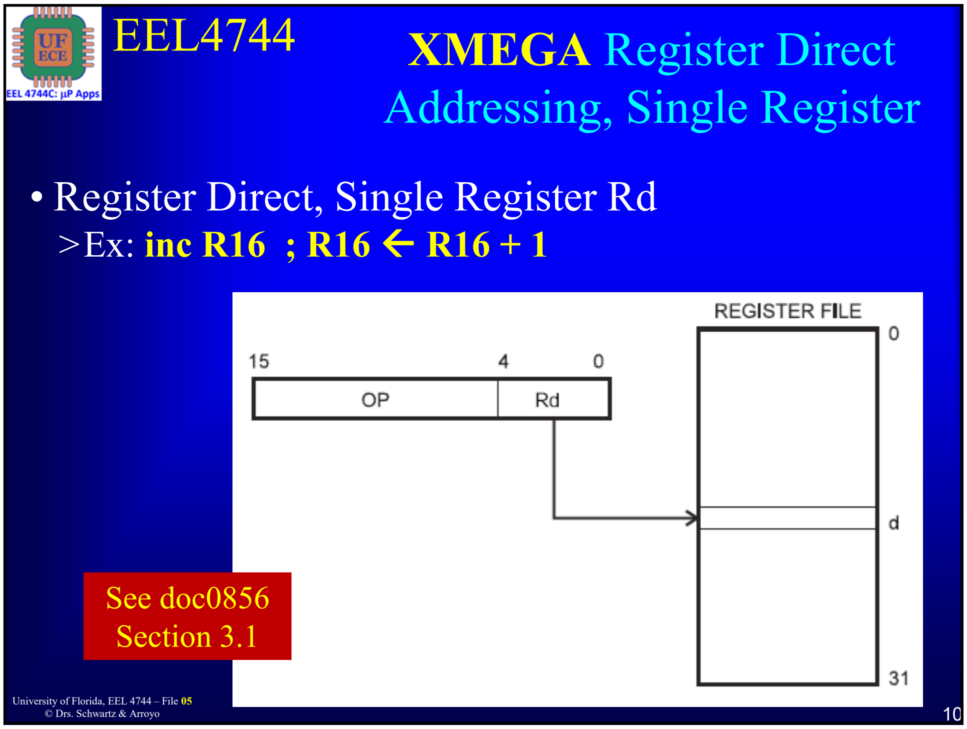
Queue



EEL4744

Addressing Modes for XMEGA

- Direct Addressing Mode
  >Rd (destination) and Rr (source) Registers, JMP, CALL, examples: **LDS, STS**
- Indirect Addressing Mode
  >X, Y, Z Registers, IJMP, ICALL
- Extended Addressing Mode
  >EIJMP, EICALL
- Constant Addressing Mode
  >LPM, SPM - load/store program memory
- Relative Addressing Mode
  >RJMP and RCALL (PC = PC + k +1)

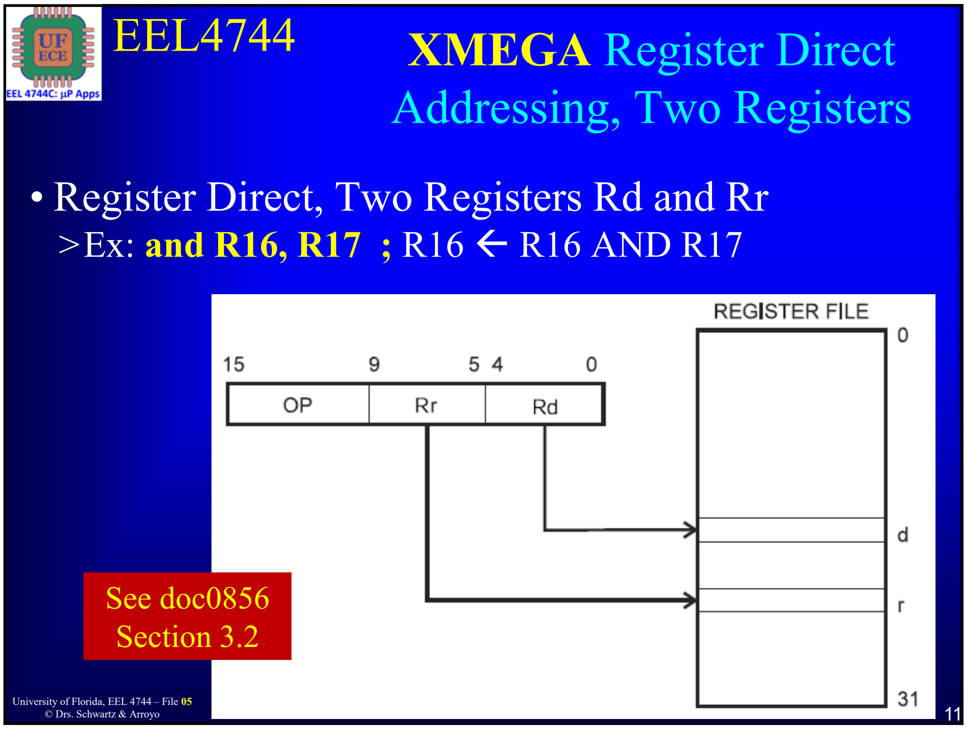


EEL4744

XMEGA Register Direct Addressing, Single Register

- Register Direct, Single Register Rd
  >Ex: **inc R16 ; R16 ← R16 + 1**

See doc0856
Section 3.1



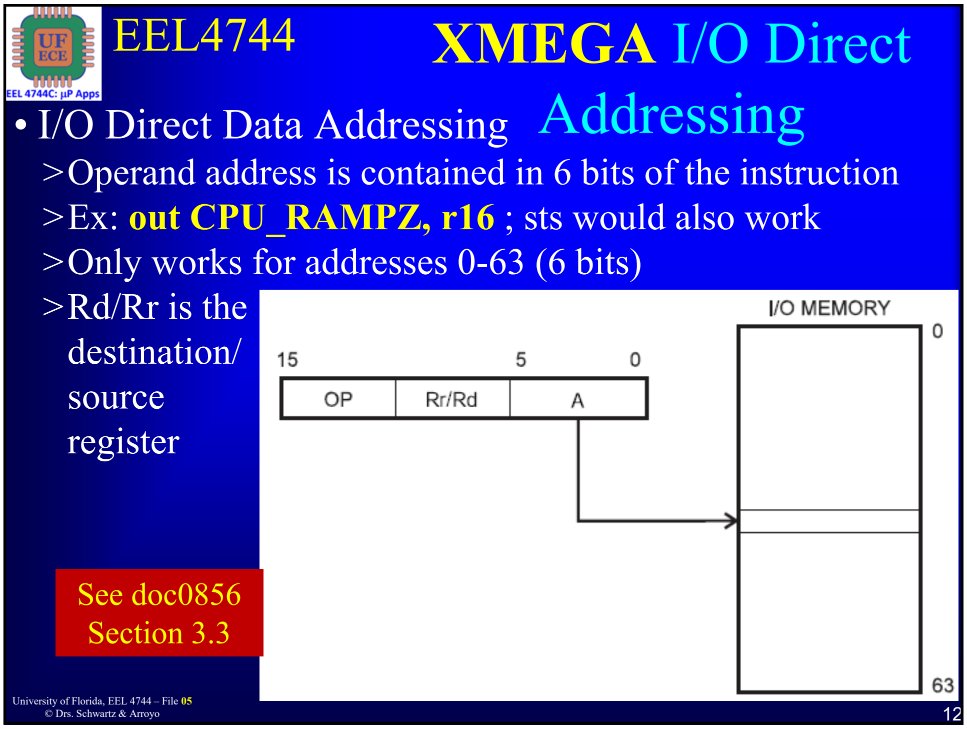
EEL4744

XMEGA Register Direct Addressing, Two Registers

- Register Direct, Two Registers Rd and Rr
  - >Ex: **and R16, R17 ;** R16 ← R16 AND R17

See doc0856 Section 3.2

11



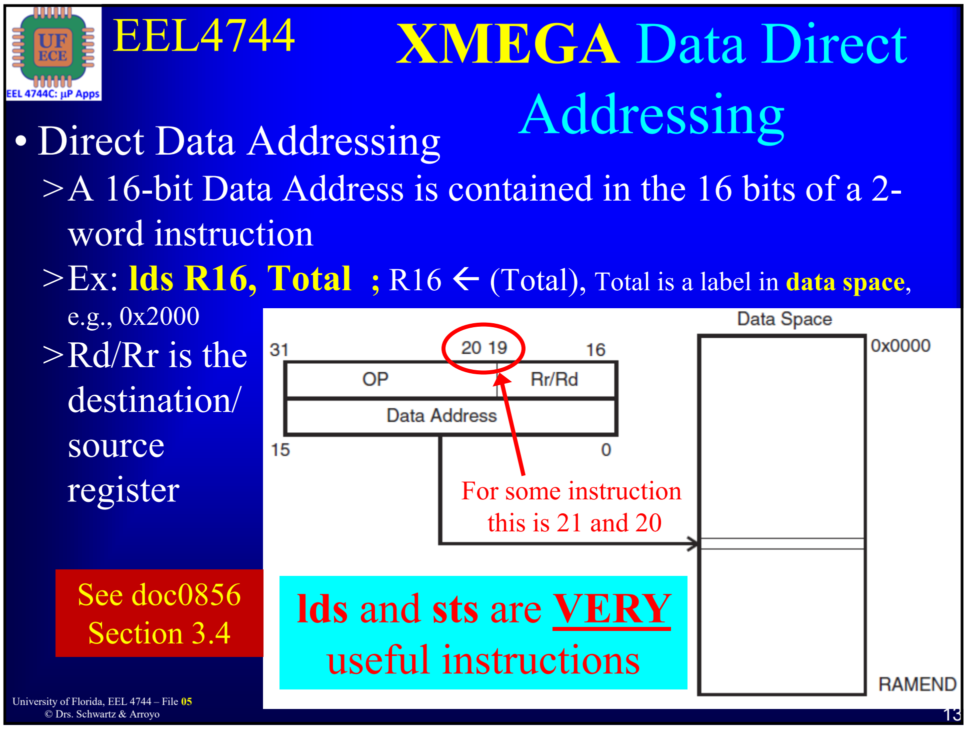
EEL4744

XMEGA I/O Direct Addressing

- I/O Direct Data Addressing
  - >Operand address is contained in 6 bits of the instruction
  - >Ex: **out CPU_RAMPZ, r16** ; sts would also work
  - >Only works for addresses 0-63 (6 bits)
  - >Rd/Rr is the destination/ source register

See doc0856 Section 3.3

12



EEL4744 **XMEGA** Data Direct Addressing

- Direct Data Addressing
  - >A 16-bit Data Address is contained in the 16 bits of a 2-word instruction
  - >Ex: **lds R16, Total** **;** R16 ← (Total), Total is a label in **data space**, e.g., 0x2000
  - >Rd/Rr is the destination/ source register

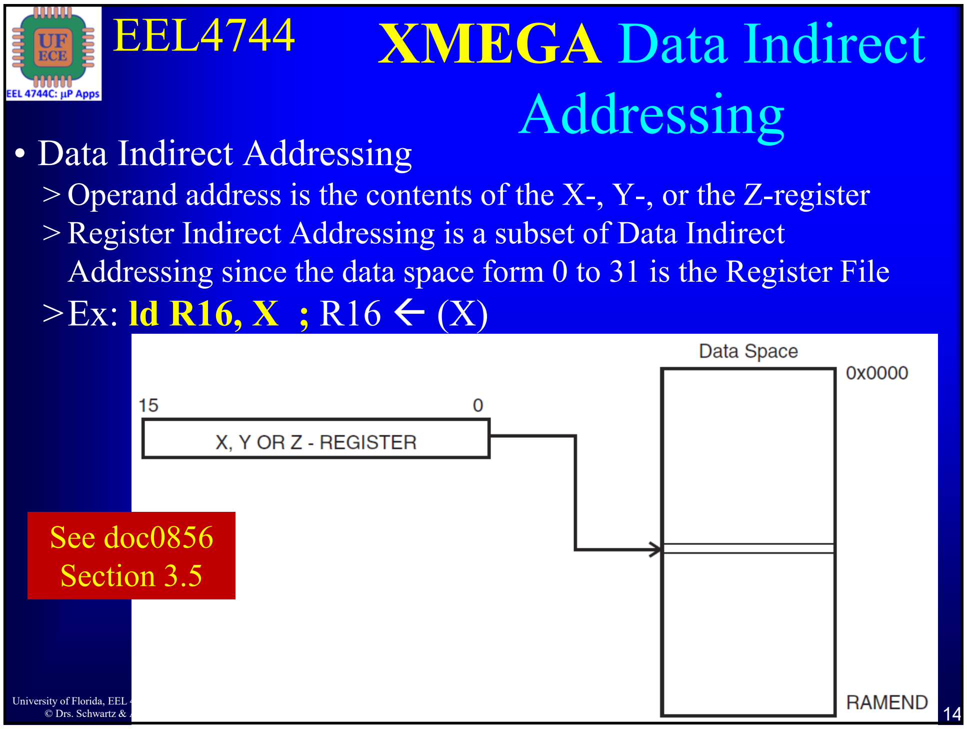
For some instruction this is 21 and 20

See doc0856 Section 3.4

lds and **sts** are **VERY** useful instructions

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo

13



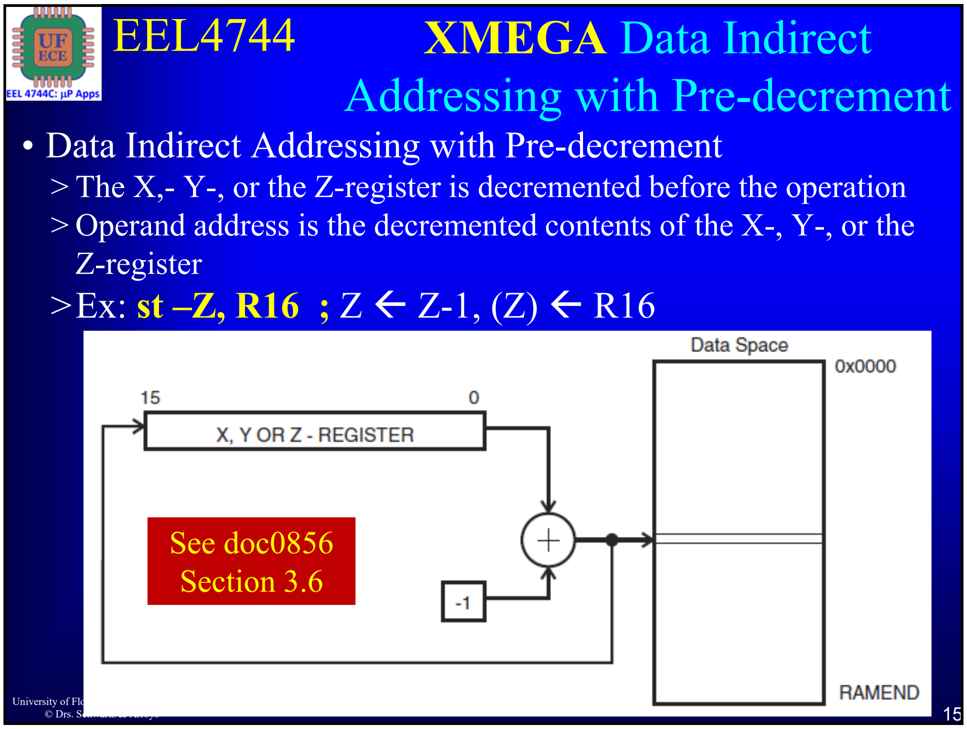
EEL4744 **XMEGA** Data Indirect Addressing

- Data Indirect Addressing
  - >Operand address is the contents of the X-, Y-, or the Z-register
  - >Register Indirect Addressing is a subset of Data Indirect Addressing since the data space form 0 to 31 is the Register File
  - >Ex: **ld R16, X** **;** R16 ← (X)

See doc0856 Section 3.5

University of Florida, EEL
© Drs. Schwartz &

14



EEL4744 **XMEGA** Data Indirect Addressing with Pre-decrement

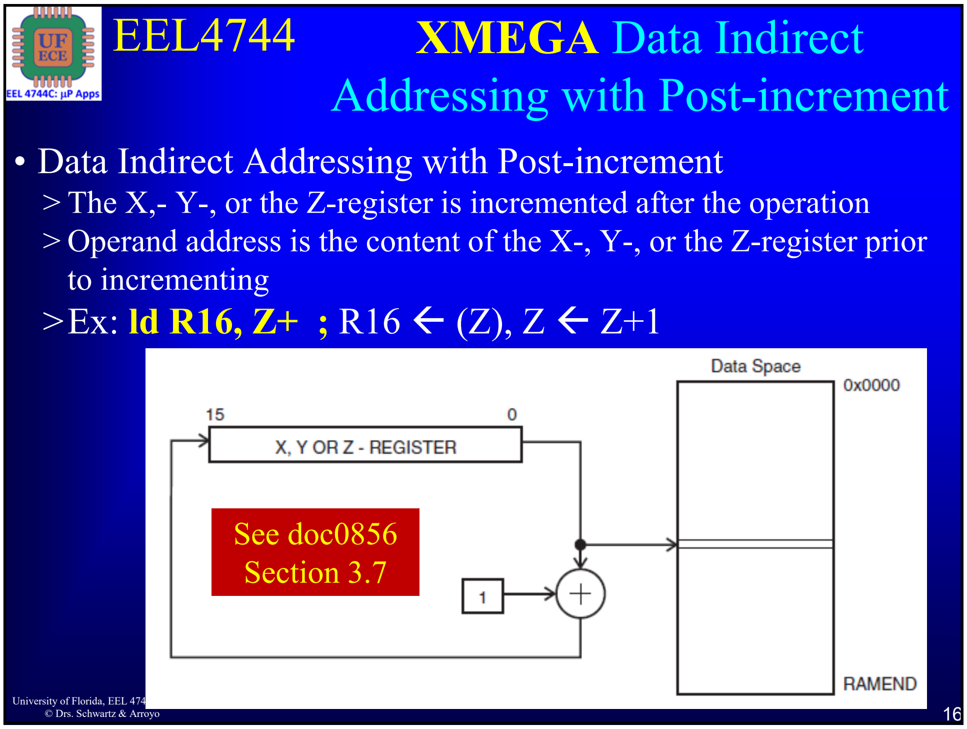
- Data Indirect Addressing with Pre-decrement
  > The X,- Y-, or the Z-register is decremented before the operation
  > Operand address is the decremented contents of the X-, Y-, or the Z-register
  > Ex: **st –Z, R16 ;** Z ← Z-1, (Z) ← R16

See doc0856 Section 3.6

University of Fl
© Drs. S

15

15



EEL4744 **XMEGA** Data Indirect Addressing with Post-increment

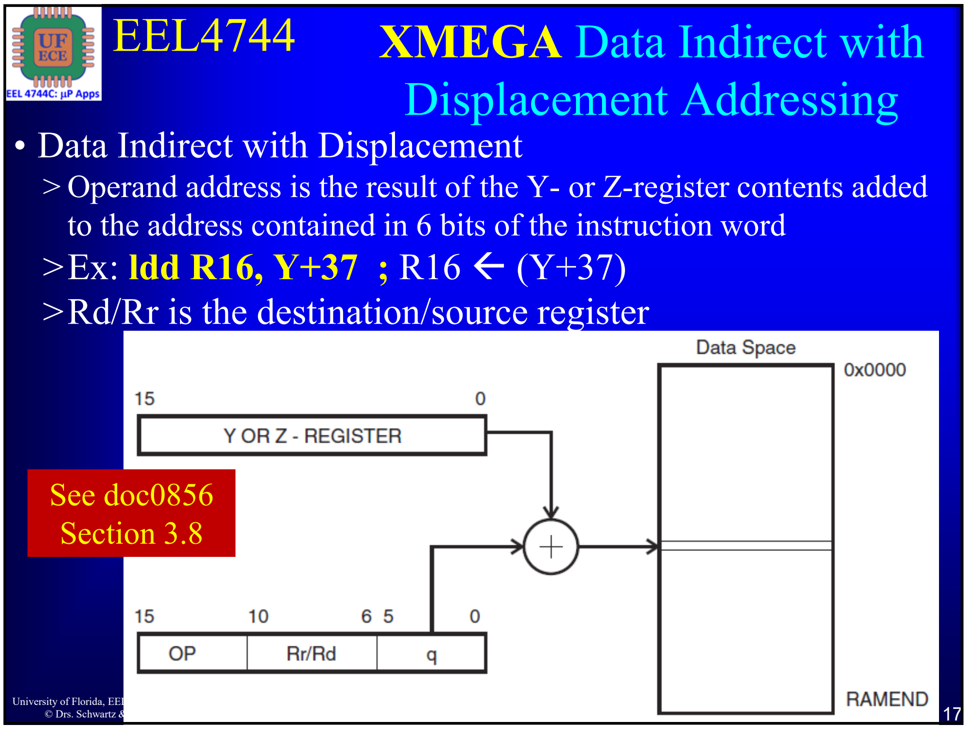
- Data Indirect Addressing with Post-increment
  > The X,- Y-, or the Z-register is incremented after the operation
  > Operand address is the content of the X-, Y-, or the Z-register prior to incrementing
  > Ex: **ld R16, Z+ ;** R16 ← (Z), Z ← Z+1

See doc0856 Section 3.7

University of Florida, EEL 474
© Drs. Schwartz & Arroyo

16

16

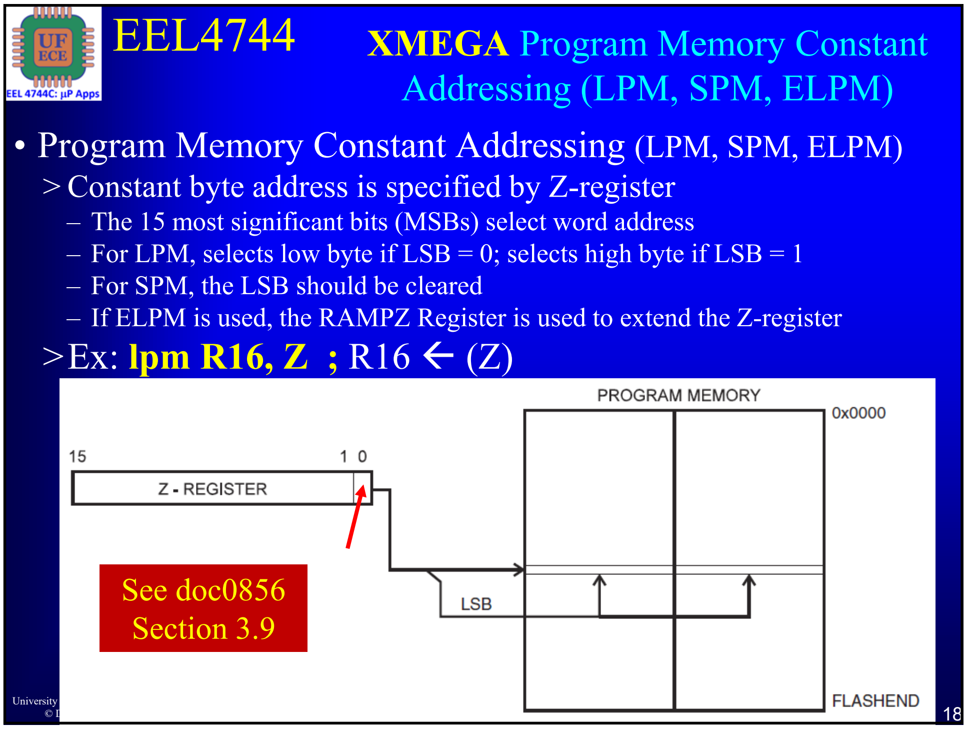


EEL4744 **XMEGA** Data Indirect with Displacement Addressing

- Data Indirect with Displacement
  - > Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word
  - >Ex: **ldd R16, Y+37 ;** R16 ← (Y+37)
  - >Rd/Rr is the destination/source register

See doc0856 Section 3.8

17

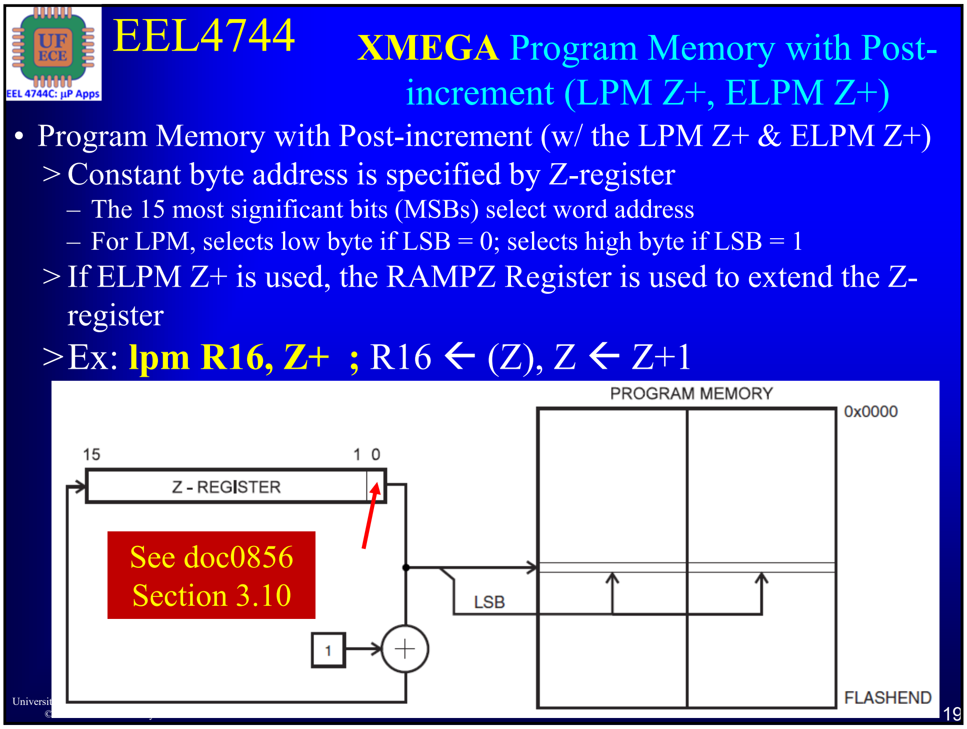


EEL4744 **XMEGA** Program Memory Constant Addressing (LPM, SPM, ELPM)

- Program Memory Constant Addressing (LPM, SPM, ELPM)
  - > Constant byte address is specified by Z-register
    - – The 15 most significant bits (MSBs) select word address
    - – For LPM, selects low byte if LSB = 0; selects high byte if LSB = 1
    - – For SPM, the LSB should be cleared
    - – If ELPM is used, the RAMPZ Register is used to extend the Z-register
  - >Ex: **lpm R16, Z ;** R16 ← (Z)

See doc0856 Section 3.9

18



EEL4744

XMEGA Program Memory with Post-increment (LPM Z+, ELPM Z+)

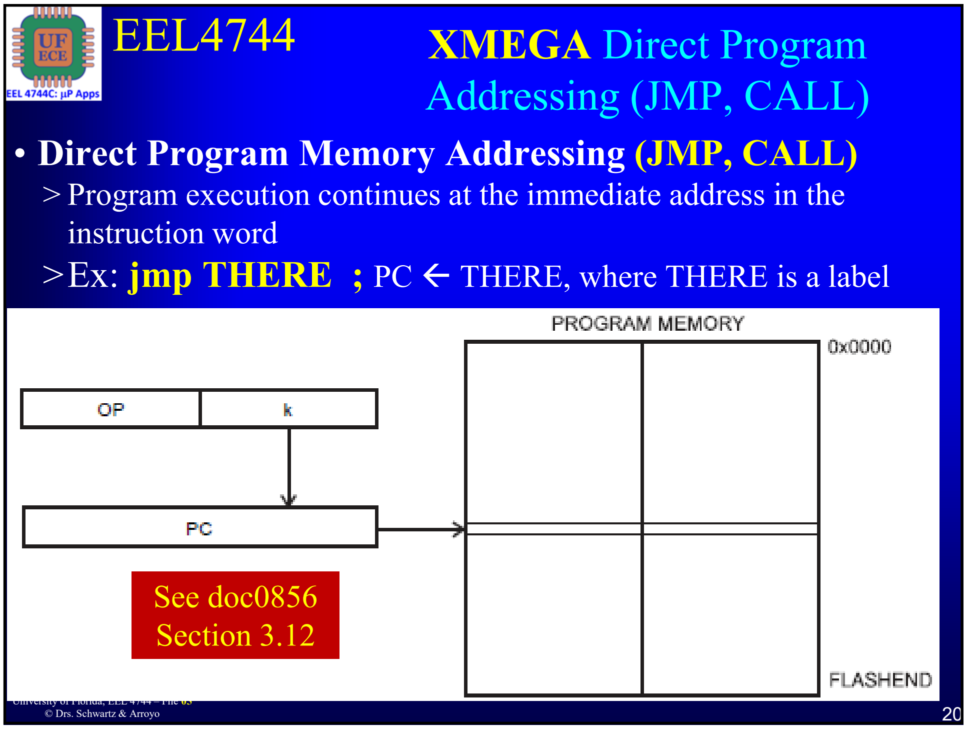
- Program Memory with Post-increment (w/ the LPM Z+ & ELPM Z+)
  - > Constant byte address is specified by Z-register
    - – The 15 most significant bits (MSBs) select word address
    - – For LPM, selects low byte if LSB = 0; selects high byte if LSB = 1
  - > If ELPM Z+ is used, the RAMPZ Register is used to extend the Z-register
  - > Ex: **lpm R16, Z+ ;** R16 ← (Z), Z ← Z+1

See doc0856 Section 3.10

Universit

19

19



EEL4744

XMEGA Direct Program Addressing (JMP, CALL)

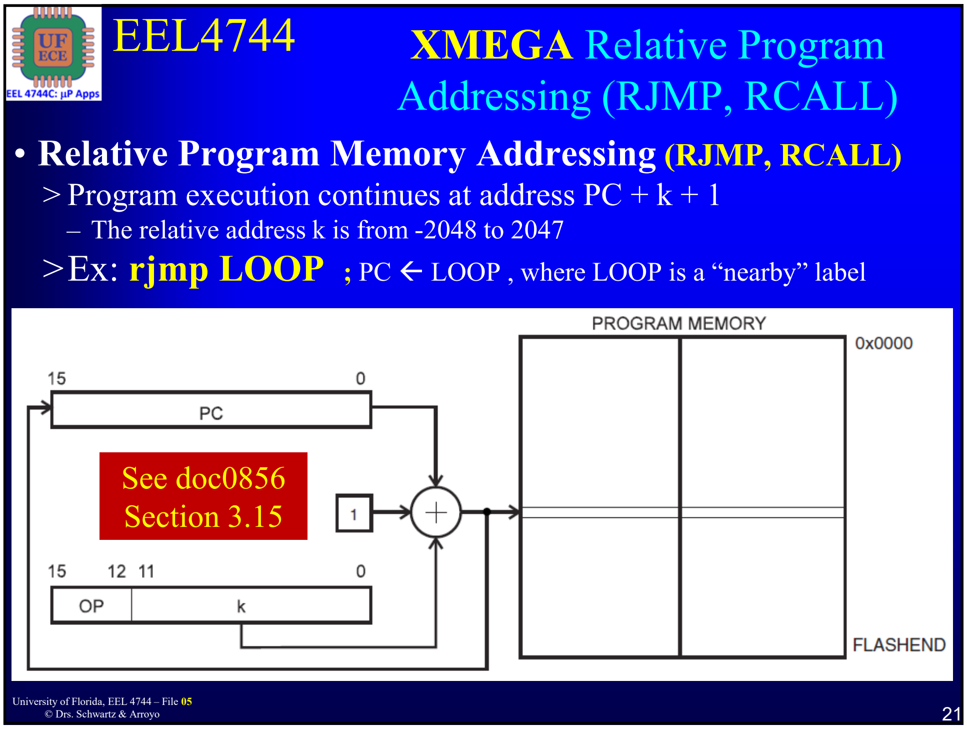
- **Direct Program Memory Addressing (JMP, CALL)**
  - > Program execution continues at the immediate address in the instruction word
  - > Ex: **jmp THERE ;** PC ← THERE, where THERE is a label

See doc0856 Section 3.12

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo

20

20



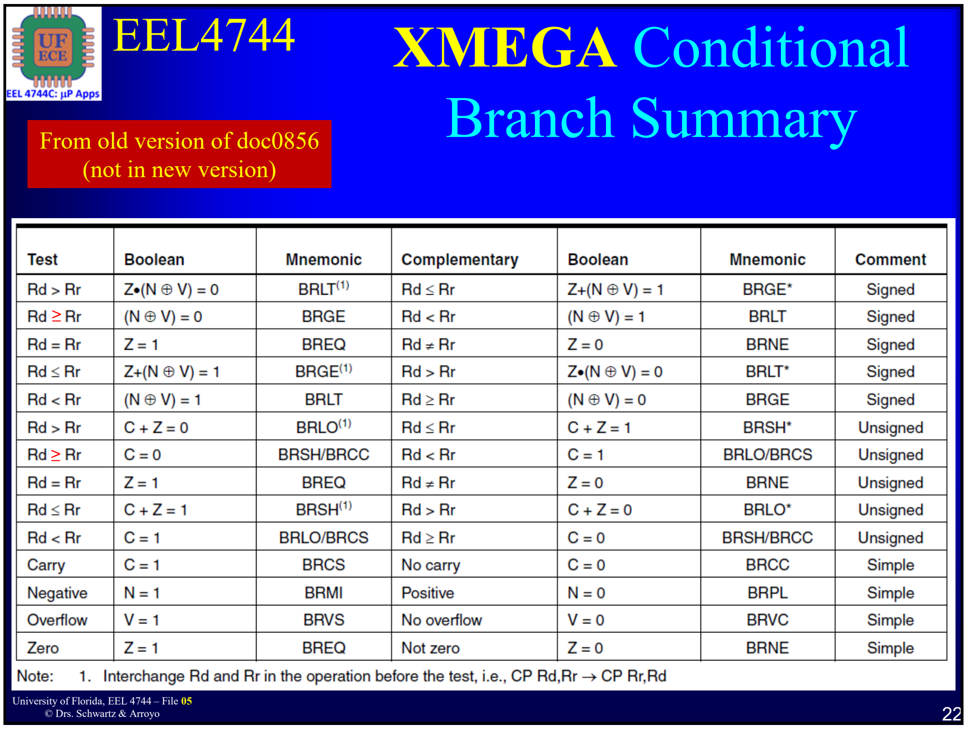
EEL4744

XMEGA Relative Program Addressing (RJMP, RCALL)

- **Relative Program Memory Addressing (RJMP, RCALL)**
  - > Program execution continues at address PC + k + 1
    - – The relative address k is from -2048 to 2047
  - > Ex: **rjmp LOOP** ; PC ← LOOP , where LOOP is a "nearby" label

See doc0856 Section 3.15

21



EEL4744

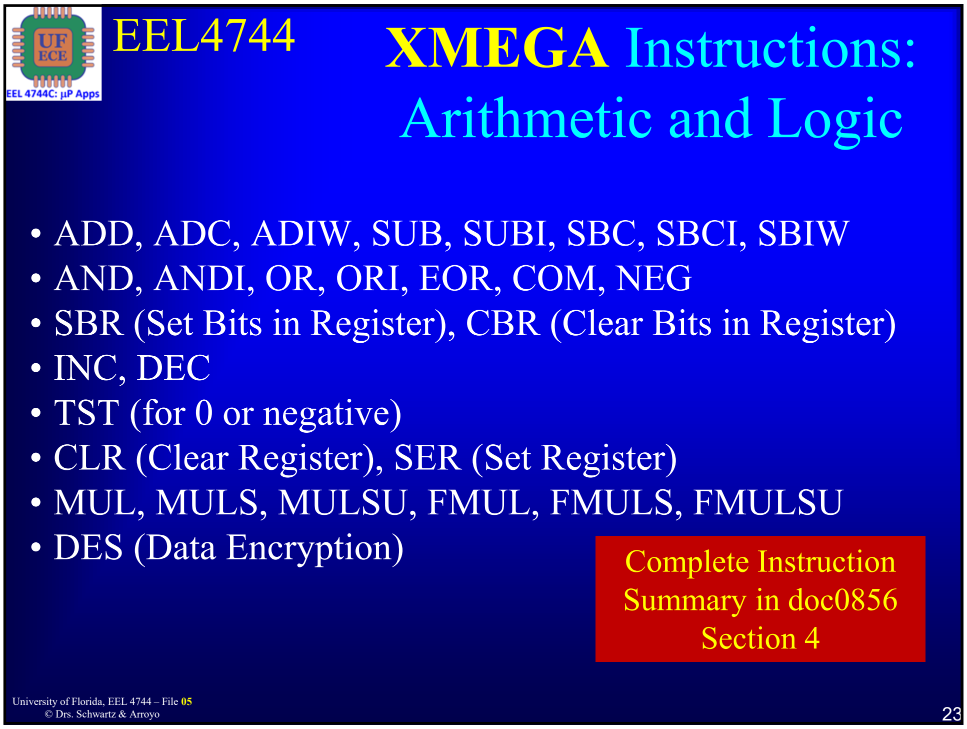
XMEGA Conditional Branch Summary

From old version of doc0856 (not in new version)

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z•(N ⊕ V) = 0	BRLT$^{(1)}$	Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Rd < Rr	(N ⊕ V) = 1	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE$^{(1)}$	Rd > Rr	Z•(N ⊕ V) = 0	BRLT*	Signed
Rd < Rr	(N ⊕ V) = 1	BRLT	Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO$^{(1)}$	Rd ≤ Rr	C + Z = 1	BRSH*	Unsigned
Rd ≥ Rr	C = 0	BRSH/BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Unsigned
Rd ≤ Rr	C + Z = 1	BRSH$^{(1)}$	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ≥ Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test, i.e., CP Rd,Rr → CP Rr,Rd

22



EEL4744

XMEGA Instructions: Arithmetic and Logic

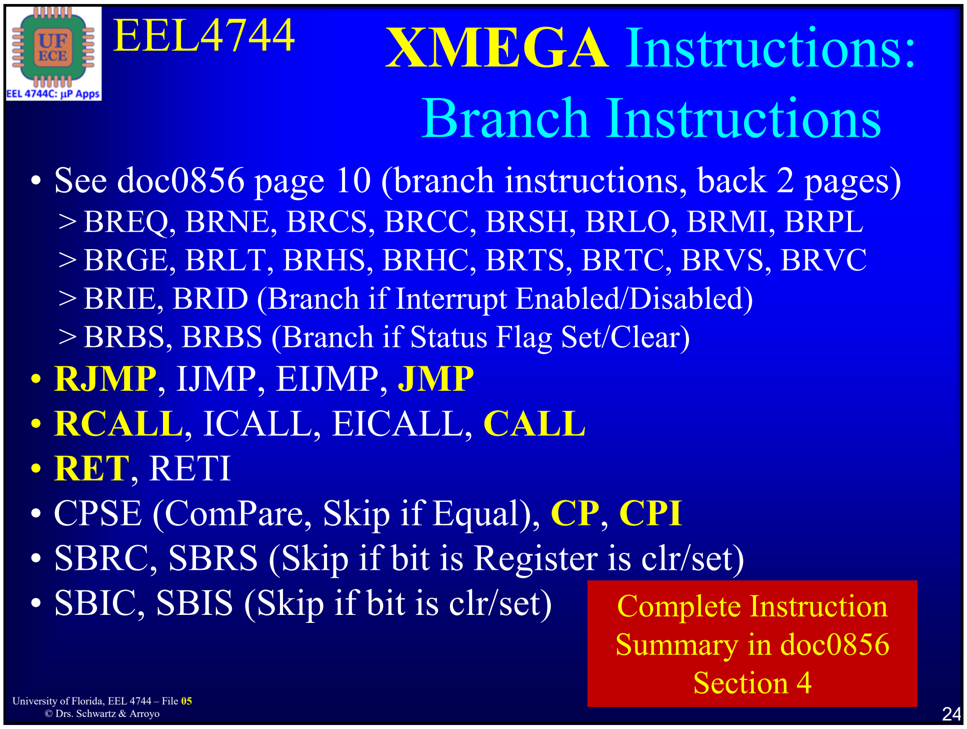
- ADD, ADC, ADIW, SUB, SUBI, SBC, SBCI, SBIW
- AND, ANDI, OR, ORI, EOR, COM, NEG
- SBR (Set Bits in Register), CBR (Clear Bits in Register)
- INC, DEC
- TST (for 0 or negative)
- CLR (Clear Register), SER (Set Register)
- MUL, MULS, MULSU, FMUL, FMULS, FMULSU
- DES (Data Encryption)

Complete Instruction Summary in doc0856 Section 4

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo

23

23



EEL4744

XMEGA Instructions: Branch Instructions

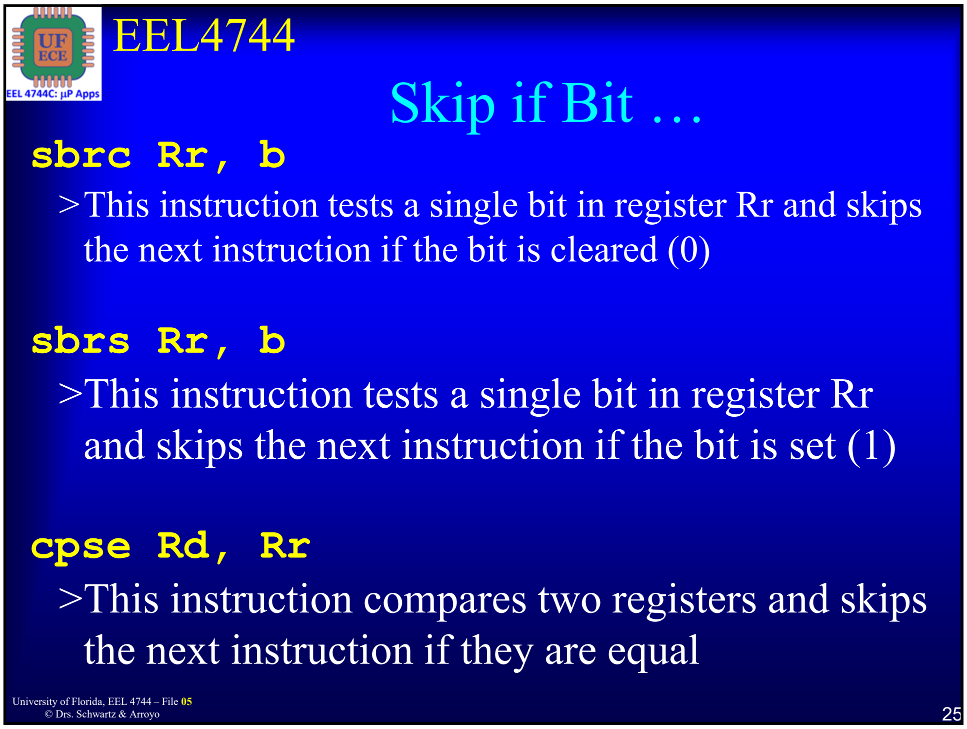
- See doc0856 page 10 (branch instructions, back 2 pages)
  - > BREQ, BRNE, BRCS, BRCC, BRSH, BRLO, BRMI, BRPL
  - > BRGE, BRLT, BRHS, BRHC, BRTS, BRTC, BRVS, BRVC
  - > BRIE, BRID (Branch if Interrupt Enabled/Disabled)
  - > BRBS, BRBS (Branch if Status Flag Set/Clear)
- **RJMP**, IJMP, EIJMP, **JMP**
- **RCALL**, ICALL, EICALL, **CALL**
- **RET**, RETI
- CPSE (ComPare, Skip if Equal), **CP**, **CPI**
- SBRC, SBRS (Skip if bit is Register is clr/set)
- SBIC, SBIS (Skip if bit is clr/set)

Complete Instruction Summary in doc0856 Section 4

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo

24

24



EEL4744

Skip if Bit …

`sbrc Rr, b`

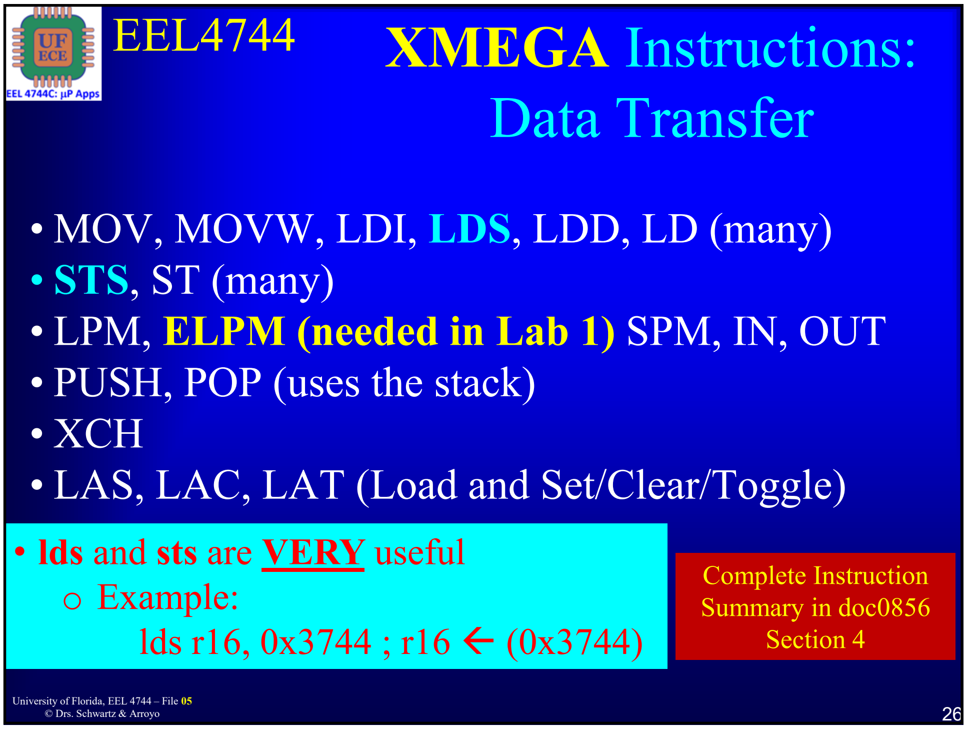
> This instruction tests a single bit in register Rr and skips the next instruction if the bit is cleared (0)

`sbrs Rr, b`

> This instruction tests a single bit in register Rr and skips the next instruction if the bit is set (1)

`cpse Rd, Rr`

> This instruction compares two registers and skips the next instruction if they are equal



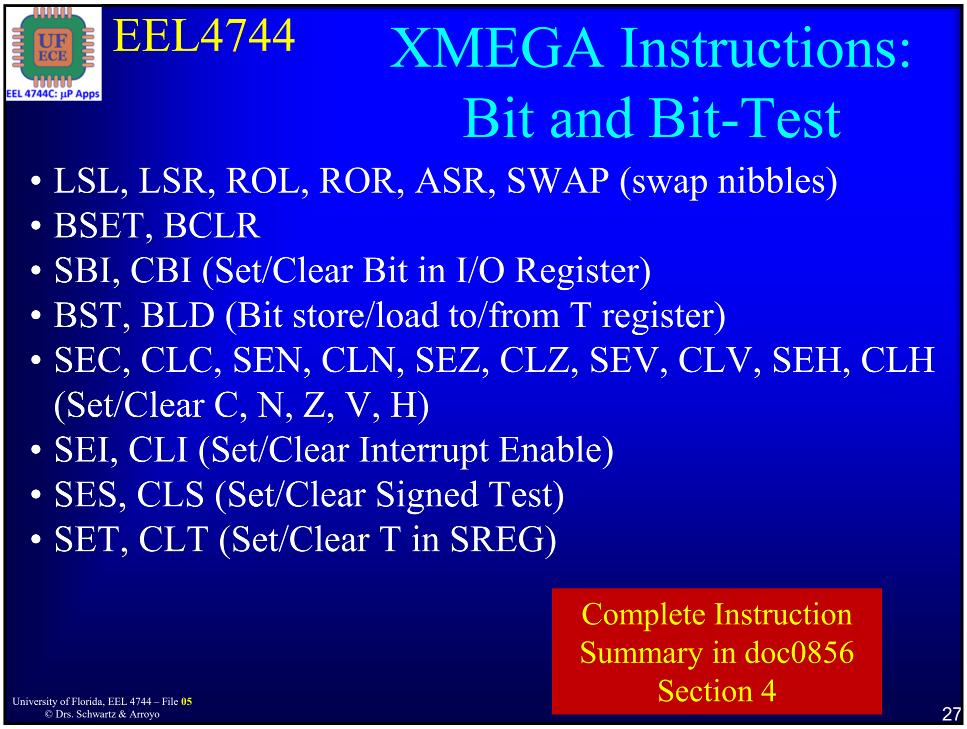
EEL4744

XMEGA Instructions: Data Transfer

- MOV, MOVW, LDI, **LDS**, LDD, LD (many)
- **STS**, ST (many)
- LPM, **ELPM (needed in Lab 1)** SPM, IN, OUT
- PUSH, POP (uses the stack)
- XCH
- LAS, LAC, LAT (Load and Set/Clear/Toggle)

- **lds** and **sts** are **VERY** useful
  - Example:
    lds r16, 0x3744 ; r16 ← (0x3744)

Complete Instruction Summary in doc0856 Section 4



EEL4744

XMEGA Instructions: Bit and Bit-Test

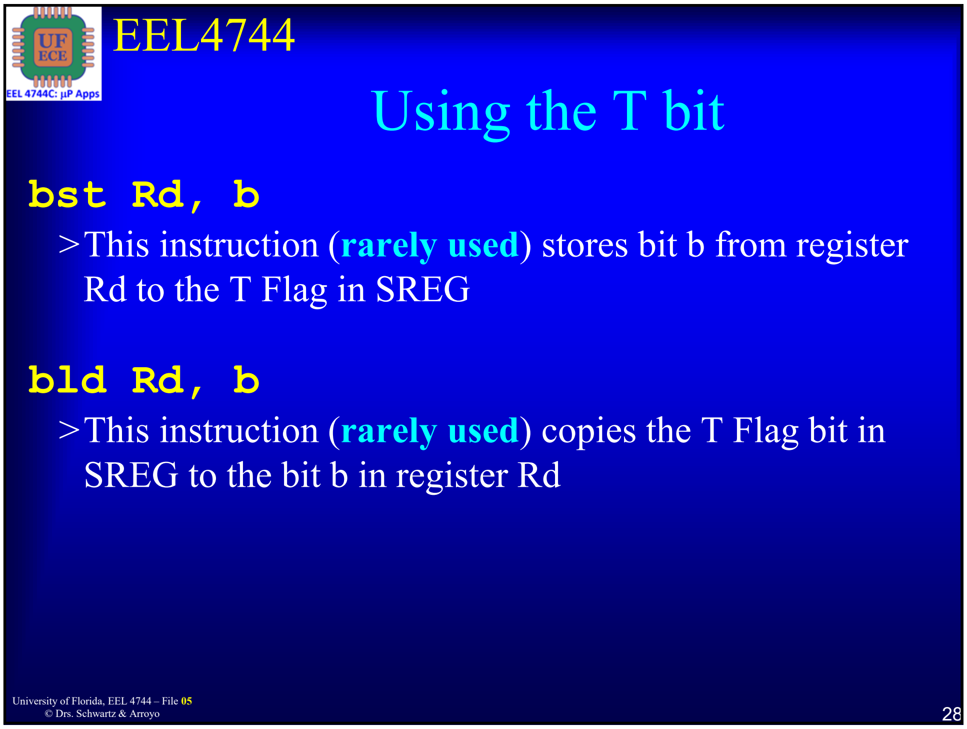
- LSL, LSR, ROL, ROR, ASR, SWAP (swap nibbles)
- BSET, BCLR
- SBI, CBI (Set/Clear Bit in I/O Register)
- BST, BLD (Bit store/load to/from T register)
- SEC, CLC, SEN, CLN, SEZ, CLZ, SEV, CLV, SEH, CLH (Set/Clear C, N, Z, V, H)
- SEI, CLI (Set/Clear Interrupt Enable)
- SES, CLS (Set/Clear Signed Test)
- SET, CLT (Set/Clear T in SREG)

Complete Instruction Summary in doc0856 Section 4

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo

27

27



EEL4744

Using the T bit

bst Rd, b

> This instruction (**rarely used**) stores bit b from register Rd to the T Flag in SREG

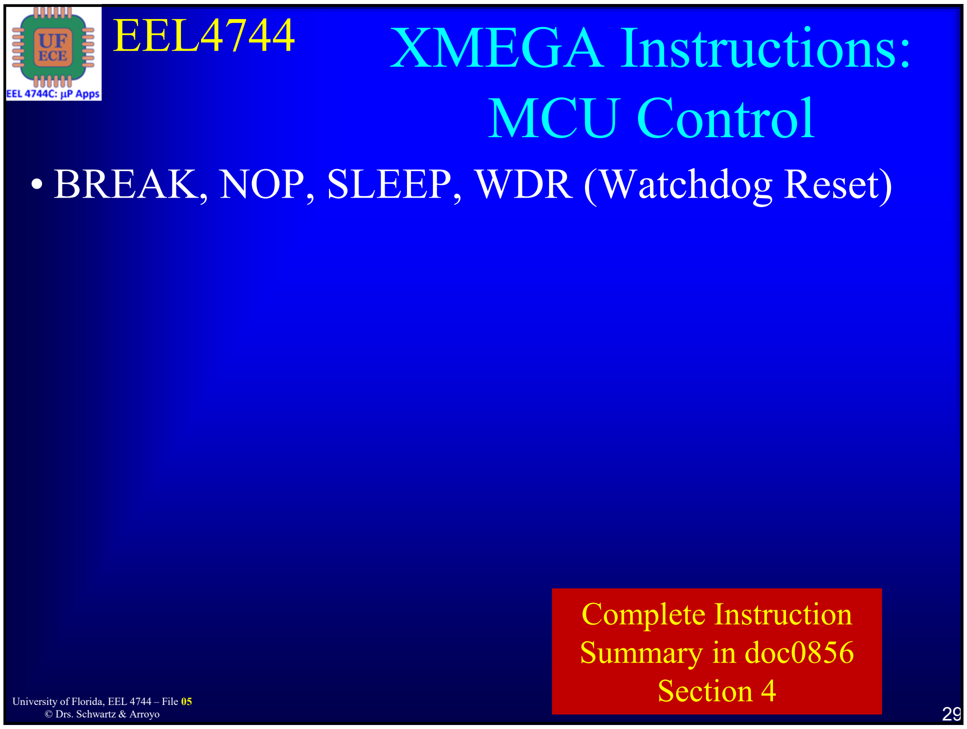
bld Rd, b

> This instruction (**rarely used**) copies the T Flag bit in SREG to the bit b in register Rd

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo

28

28

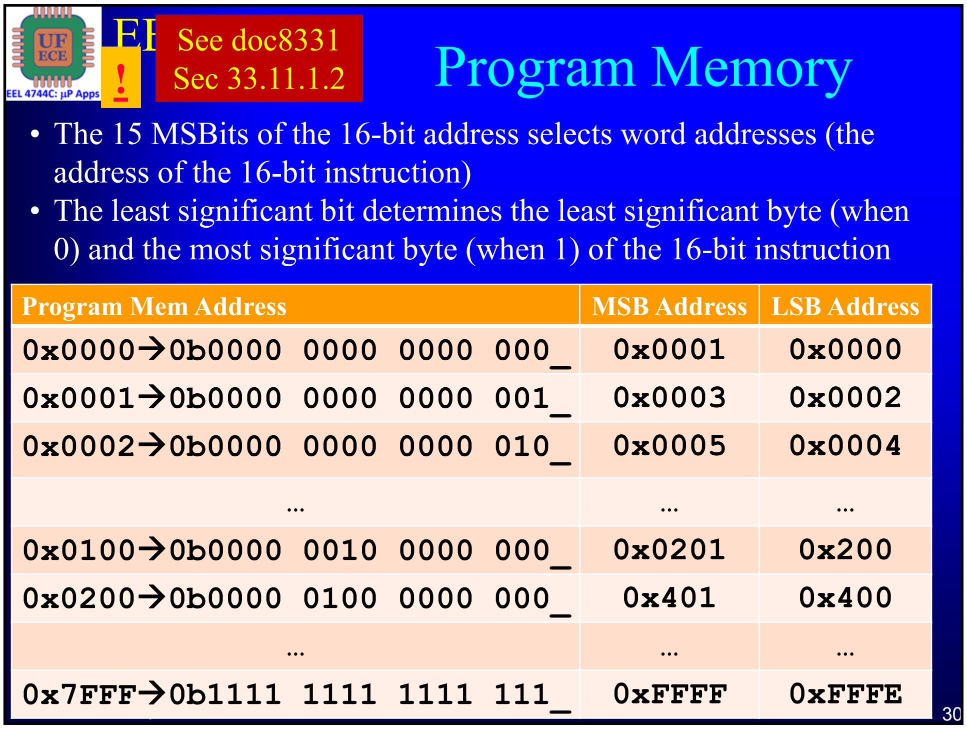


EEL4744

XMEGA Instructions: MCU Control

- BREAK, NOP, SLEEP, WDR (Watchdog Reset)

Complete Instruction Summary in doc0856 Section 4



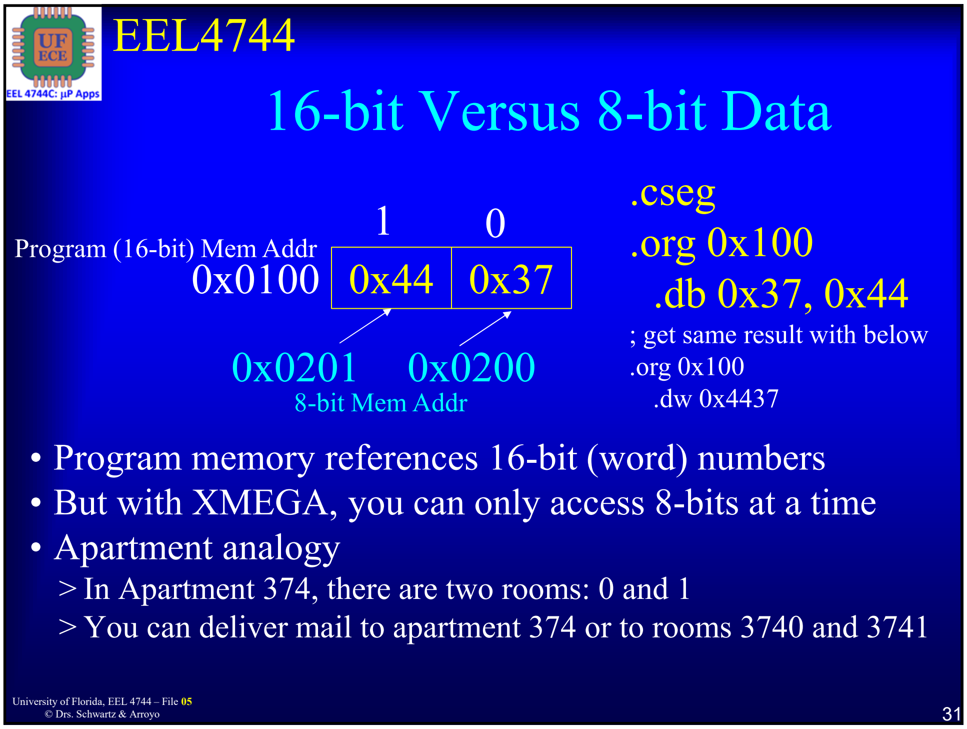
EE

See doc8331 Sec 33.11.1.2

Program Memory

- The 15 MSBits of the 16-bit address selects word addresses (the address of the 16-bit instruction)
- The least significant bit determines the least significant byte (when 0) and the most significant byte (when 1) of the 16-bit instruction

Program Mem Address	MSB Address	LSB Address
0x0000→0b0000 0000 0000 000_	0x0001	0x0000
0x0001→0b0000 0000 0000 001_	0x0003	0x0002
0x0002→0b0000 0000 0000 010_	0x0005	0x0004
…	…	…
0x0100→0b0000 0010 0000 000_	0x0201	0x200
0x0200→0b0000 0100 0000 000_	0x401	0x400
…	…	…
0x7FFF→0b1111 1111 1111 111_	0xFFFF	0xFFFE



EEL4744

16-bit Versus 8-bit Data

Program (16-bit) Mem Addr

0x0100

1	0
0x44	0x37
0x0201	0x0200

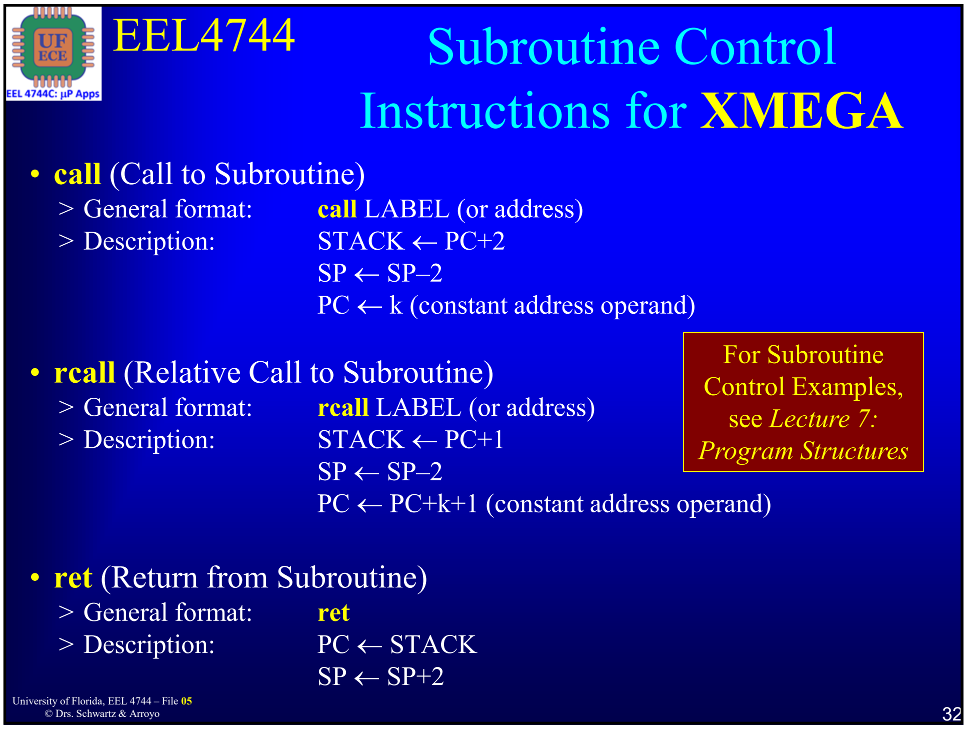
8-bit Mem Addr

```
.cseg
.org 0x100
  .db 0x37, 0x44
; get same result with below
.org 0x100
  .dw 0x4437
```

- Program memory references 16-bit (word) numbers
- But with XMEGA, you can only access 8-bits at a time
- Apartment analogy
  - > In Apartment 374, there are two rooms: 0 and 1
  - > You can deliver mail to apartment 374 or to rooms 3740 and 3741

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo 31

31



EEL4744

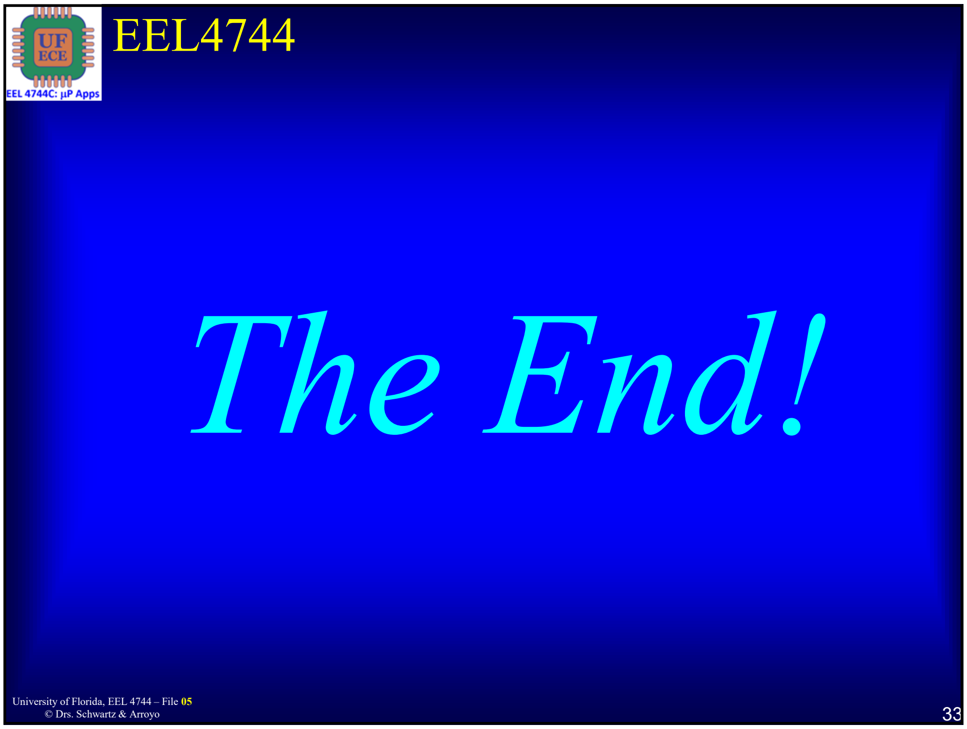
Subroutine Control Instructions for **XMEGA**

- **call** (Call to Subroutine)
  - > General format: **call** LABEL (or address)
  - > Description: STACK ← PC+2
SP ← SP–2
PC ← k (constant address operand)
- **rcall** (Relative Call to Subroutine)
  - > General format: **rcall** LABEL (or address)
  - > Description: STACK ← PC+1
SP ← SP–2
PC ← PC+k+1 (constant address operand)
- **ret** (Return from Subroutine)
  - > General format: **ret**
  - > Description: PC ← STACK
SP ← SP+2

For Subroutine Control Examples, see *Lecture 7: Program Structures*

University of Florida, EEL 4744 – File 05
© Drs. Schwartz & Arroyo 32

32



EEL4744

The End!

University of Florida, EEL 4744 – File **05**
© Drs. Schwartz & Arroyo

33

33