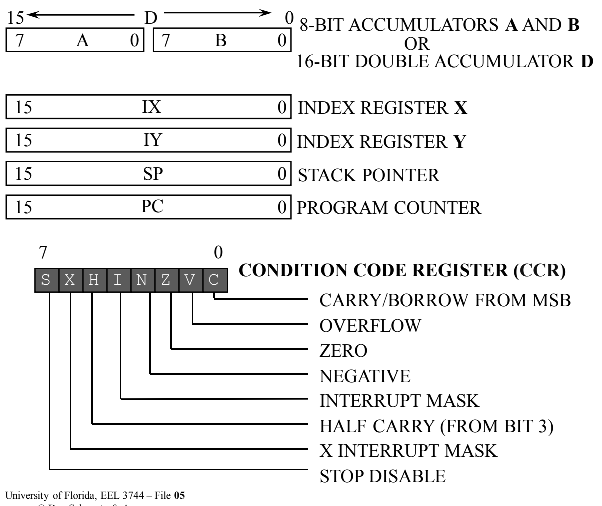
EEL 3744 Menu

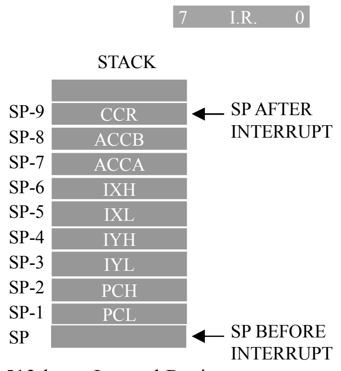
- Programming Models for the Atmel XMEGA Architecture (and others devices)
- Assembly Programming
- Addressing Modes for the XMEGA
- Instruction Set

See examples on web-site: `Imm_Addr.asm, Indirect_Addr.asm, Extended_Addr.asm, doc8331, doc0856`

68HC11/12: `Phone.asm, DirAddr.asm, ExtAddr.asm, sub_cntr.asm`

EEL 3744 **6812** Programming Model

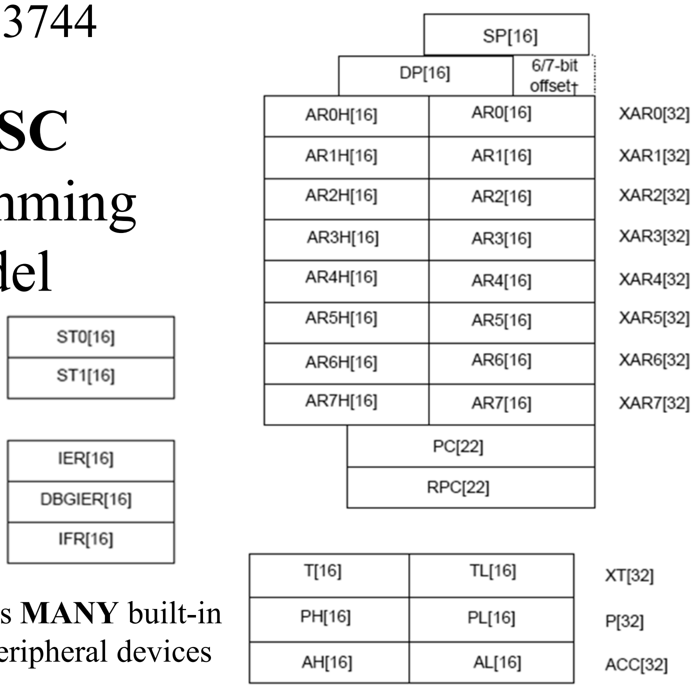




- 512-bytes Internal Registers:
  - > $0000-$00FF and $0100-$01FF
- 1k-bytes Internal SRAM:
  - > $800-$BFF
- 712-bytes Internal EEPROM
  - > $D00-$FFF
- 32k-bytes Internal Flash (single chip) or External Device (expanded)
  - > $8000-$FFFF

EEL 3744

TI **DSC** Programming Model



See spru430, Fig 2-3

See also lecture 3

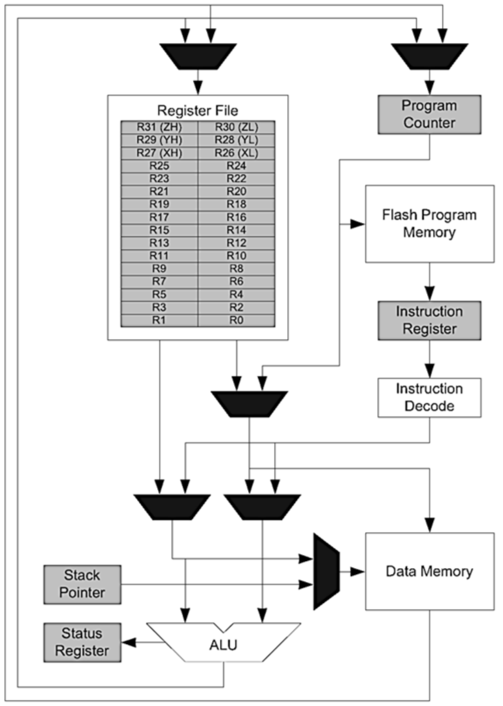
Plus **MANY** built-in peripheral devices

University of Florida, EEL 3744 – File **05**
© Drs. Schwartz & Arroyo

3

EEL 3744

Conceptual XMEGA CPU Block Diagram



See doc8331 Fig 3-1

University of Florida, EEL 3744 – File **05**
© Drs. Schwartz & Arroyo

4

EEL 3744 **6811 & 6812** General-Purpose Registers

- Within the 68HC11/12, there are two general-purpose registers.
  - > They are referred to as 8-bit registers **A** and **B** or alternatively as a 16-bit register **D**.
- Registers A and B, often called ***accumulators***

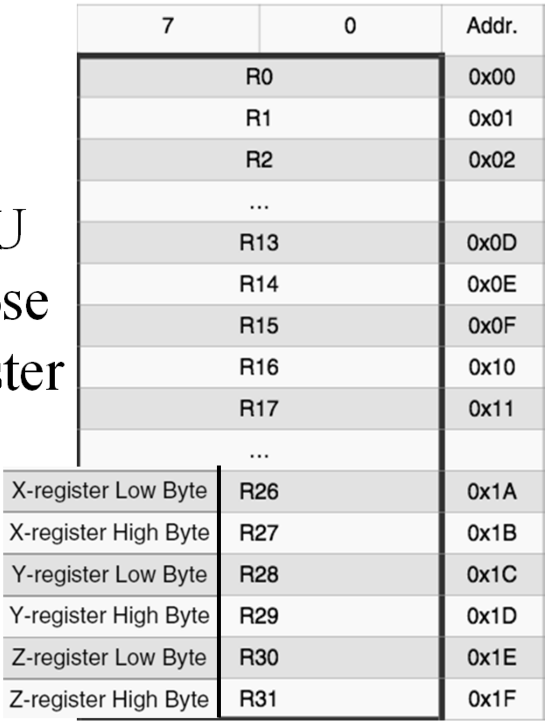
[Examples]

```
LDAA    VALUE1    ; Move the byte at location VALUE1 to Register A.
ABA               ; Add the byte in B to A; put the result in A.
LDD     WORD1     ; The 16-bit word at location WORD1 and WORD1+1 are
*                 ; moved to Register D = A | B.
ADDD    WORD2     ; The 16-bit data at Word1 and Word1+1 are added to D => D.
```

- This instruction set is ***nearly symmetric.***

[Examples] LDA (LDAA and LDAB), STA (STAA and STAB), ROL (ROLA and ROLB), etc.

EEL 3744 XMEGA CPU General Purpose Working Register Summary



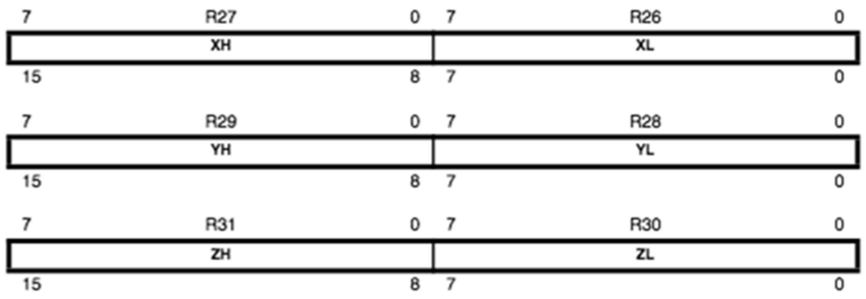
	7	0	Addr.
	R0		0x00
	R1		0x01
	R2		0x02
	...		
	R13		0x0D
	R14		0x0E
	R15		0x0F
	R16		0x10
	R17		0x11
	...		
X-register Low Byte	R26		0x1A
X-register High Byte	R27		0x1B
Y-register Low Byte	R28		0x1C
Y-register High Byte	R29		0x1D
Z-register Low Byte	R30		0x1E
Z-register High Byte	R31		0x1F

See doc8331 Fig 3-4

EEL 3744 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.



EEL 3744 XMEGA Status Register (SREG)

- 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) [**<u>actual</u>** sign of result]
V = Two's Complement Overflow Flag
N = Negative Flag
Z = Zero Flag
C = Carry Flag

See doc8331 Section 3.14.9

EEL 3744 **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 #).

EEL 3744 **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

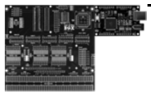
Zero (Z): Set if the result of the last arithmetic, logic, or data manipulation operation was zero; otherwise, it is reset

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

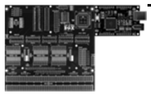
⇔:	Flag affected by instruction
0:	Flag cleared by instruction
1:	Flag set by instruction
-:	Flag not affected by instruction

See doc0856 Page 2



EEL 3744 XMEGA I/O Registers

- **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.
- **RAMPD**
  - > Register concatenated with the Z-register enabling direct addressing of the whole data space on MCUs with more than 64K bytes data space.
- **EIND**
  - > Register concatenated with the Z-register enabling indirect jump and call to the whole program space on MCUs with more than 64K words (128K bytes) program space.
- **Stack**
  - > **STACK**: Stack for return address and pushed/popped registers
  - > **SP**: Stack Pointer to STACK



EEL 3744 **6812** Special-Purpose Registers: CCR & IX/IY

❖ **Index Registers (IX and IY):** The 16-bit registers used to store the index value for operands retrieved using the indexed addressing mode.

❖ **Condition Code Register (CCR):** An 8-bit flag register in which condition codes (binary flags) are stored and tested.

S	X	H	I	N	Z	V	C

CONDITION CODE REGISTER

S - Stop Disable	N - Negative
X - X Interrupt Mask	Z - Zero
H - Half Carry	V - Arithmetic Overflow
I - I Interrupt Mask	C - Carry/Borrow

EEL 3744

6812 Flags in CCR

There are five condition flags associated with the execution of the arithmetic instructions of the M68HC11/12.

Half-Carry Flag (H): Set (to 1) if a carry occurs between bits 3 and 4 during ADD, ABA, or ADC instructions; otherwise, it is reset (to 0).

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.

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 #).

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.

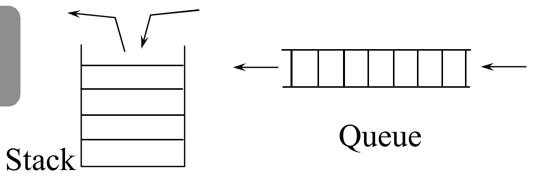
EEL 3744

68HC11/12 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



EEL 3744 Addressing Modes for **68HC11/12**

- Immediate Addressing Mode
- Direct Addressing Mode
- Extended Addressing Mode
- Indexed Addressing Mode
- Inherent Addressing Mode
- Relative Addressing Mode

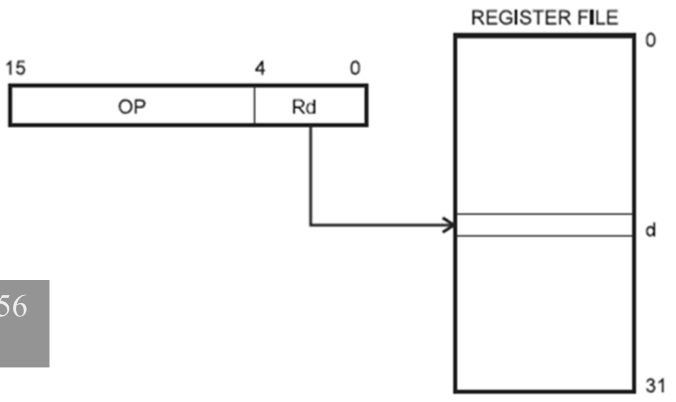
EEL 3744 Addressing Modes for **XMEGA**

- Direct Addressing Mode
  >Rd (destination) and Rr (source) Registers, JMP, CALL
- 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)

See doc0856
Page 2

EEL 3744 **XMEGA** Direct Addressing, Single Register

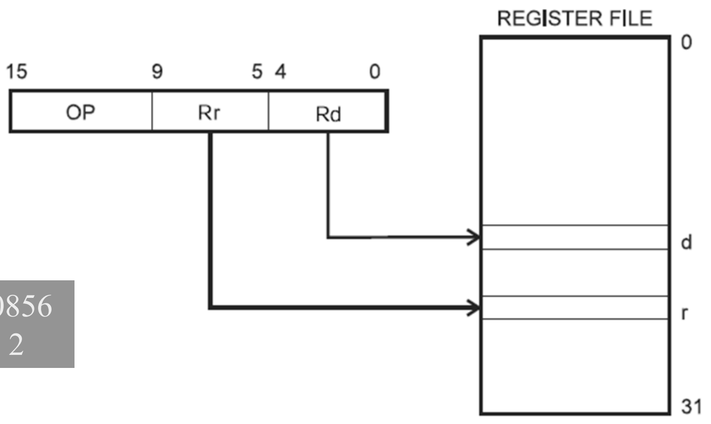
- Register Direct, Single Register Rd
  - >Ex: **inc R16 ; R16 ← R16 + 1**



See doc0856 Figure 1

EEL 3744 **XMEGA** Direct Addressing, Two Registers

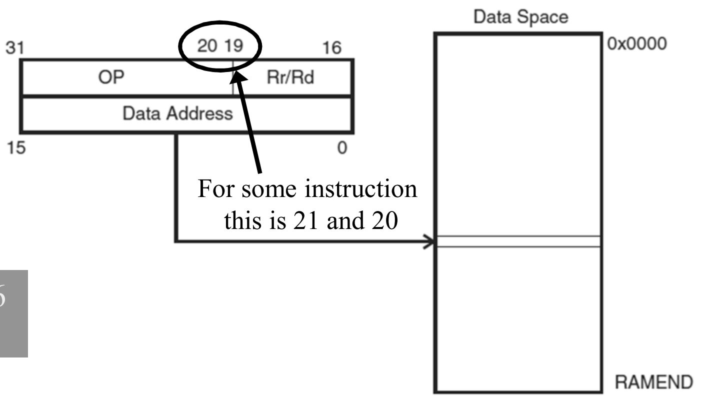
- Register Direct, Two Registers Rd and Rr
  - >Ex: **and R16, R17 ;** R16 ← R16 AND R17



See doc0856 Figure 2

EEL 3744 **XMEGA** Data Direct Addressing

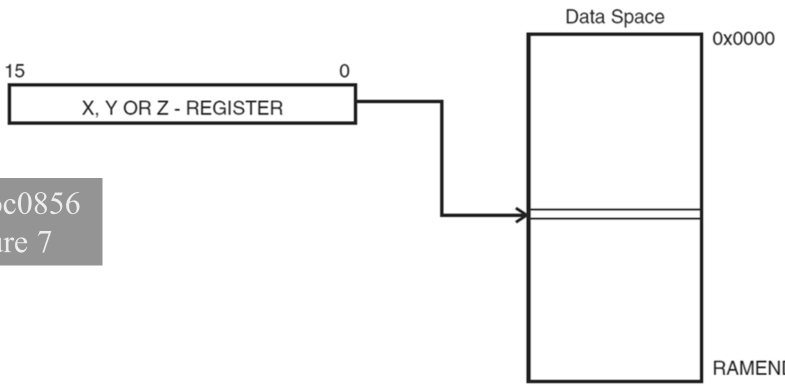
- Direct Data Addressing
  - >A 16-bit Data Address is contained in the 16 bits of a 2-word instruction
  - >Ex: **lds R16, Total** **;** R16 ← (Total), Total is a label in **data space**, e.g., 0x2000
  - >Rd/Rr is the destination/ source register



See doc0856 Figure 4

EEL 3744 **XMEGA** Data Indirect Addressing

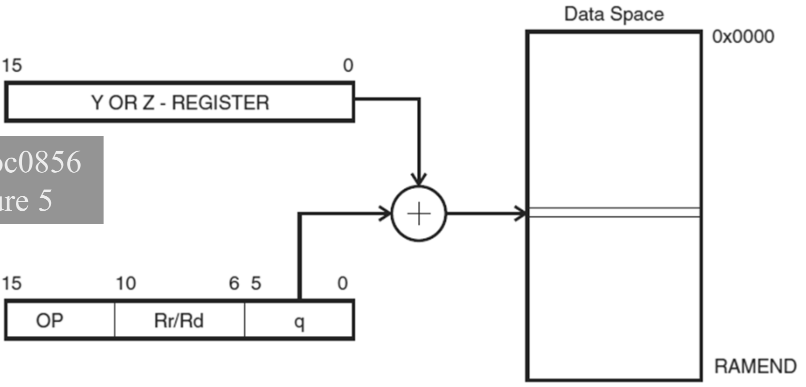
- Data Indirect Addressing
  - > Operand address is the contents of the X-, Y-, or the Z-register
  - > Register Indirect Addressing is a subset of Data Indirect Addressing since the data space form 0 to 31 is the Register File
  - >Ex: **ld R16, X** **;** R16 ← (X)



See doc0856 Figure 7

EEL 3744 **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



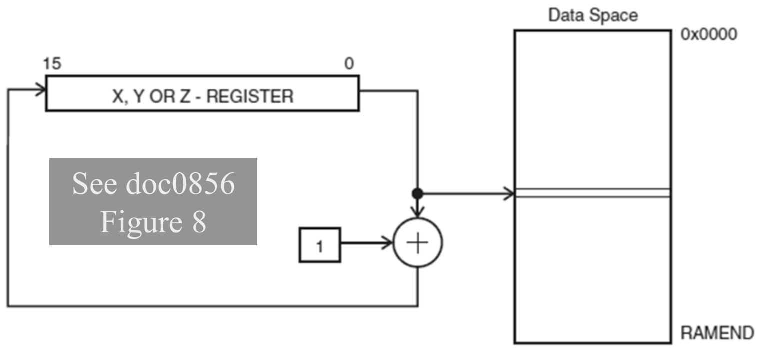
See doc0856 Figure 5

University of Florida, EEL
© Drs. Schwartz &

21

EEL 3744 **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



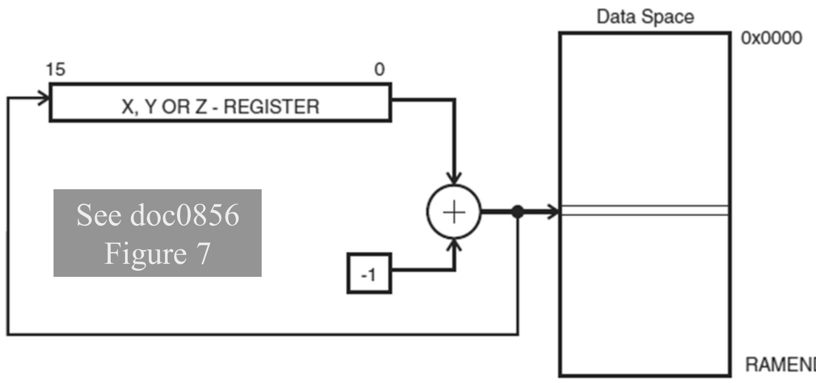
See doc0856 Figure 8

University of Florida, EEL 374
© Drs. Schwartz & Arroyo

22

EEL 3744 **XMEGA** Data Indirect Addressing with Pre-decrement

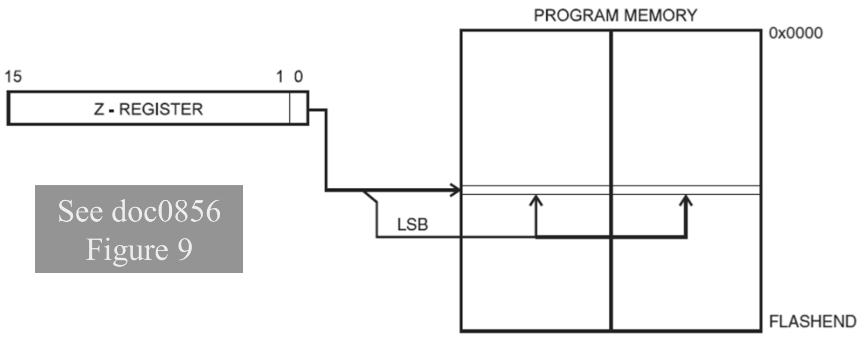
- Data Indirect Addressing with Pre-decrement
  - > The X,- Y-, or the Z-register is decremented before the operation
  - > Operand address is the decremented contents of the X-, Y-, or the Z-register
  - >Ex: **st –Z, R16 ;** Z ← Z-1, (Z) ← R16



See doc0856 Figure 7

EEL 3744 **XMEGA** Program Memory Constant Addressing (LPM, SPM, ELPM)

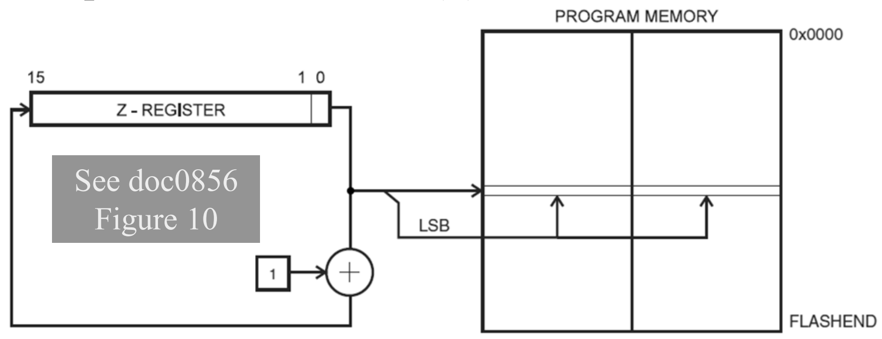
- Program Memory Constant Addressing (LPM, SPM, ELPM)
  - > Constant byte address is specified by Z-register
    - The 15 most significant bits (MSBs) select word address
    - For LPM, selects low byte if LSB = 0; selects high byte if LSB = 1
    - For SPM, the LSB should be cleared
    - If ELPM is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



See doc0856 Figure 9

EEL 3744 **XMEGA** Program Memory with Post-increment (LPM Z+, ELPM Z+)

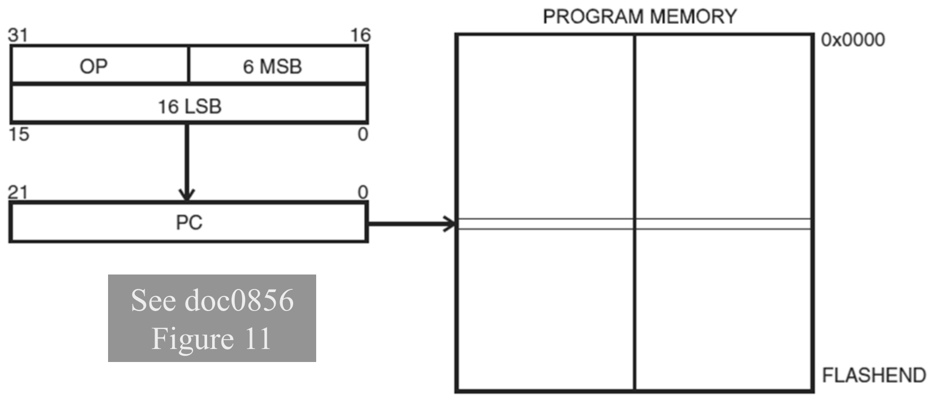
- Program Memory with Post-increment (w/ the LPM Z+ & ELPM Z+)
  - > Constant byte address is specified by Z-register
    - – The 15 most significant bits (MSBs) select word address
    - – For LPM, selects low byte if LSB = 0; selects high byte if LSB = 1
  - > If ELPM Z+ is used, the RAMPZ Register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



See doc0856 Figure 10

EEL 3744 **XMEGA** Direct Program Addressing (JMP, CALL)

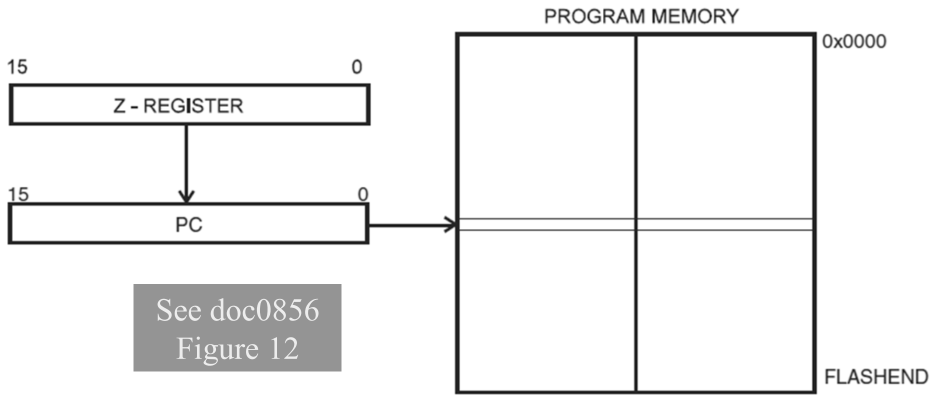
- **Direct Program Memory Addressing (JMP, CALL)**
  - > Program execution continues at the immediate address in the instruction word
  - > Ex: **jmp THERE ;** PC ← THERE, where THERE is a label



See doc0856 Figure 11

EEL 3744 **XMEGA** Indirect Program Addressing, (IJMP, ICALL)

- **Indirect Program Memory Addressing (IJMP, ICALL)**
  - > Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register)
  - >Ex: **icall ;** PC(15:0) ← Z, PC(21:16) ← 0



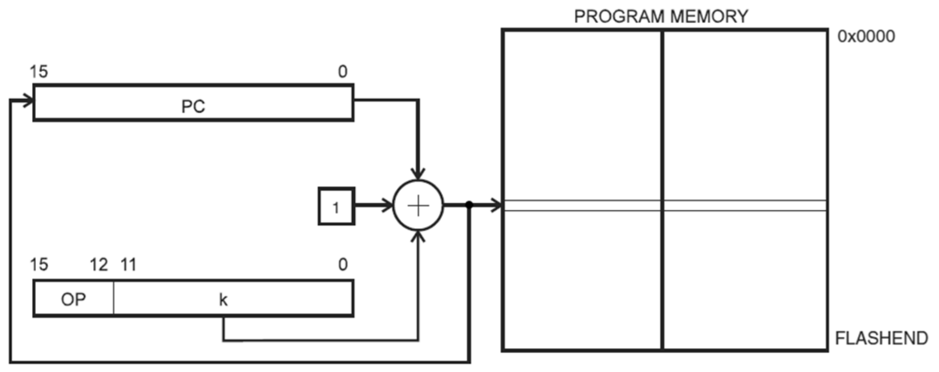
See doc0856 Figure 12

University of Florida, EEL 3744 – File 05
© Drs. Schwartz & Arroyo
27

EEL 3744 **XMEGA** Relative Program Addressing (RJMP, RCALL)

See doc0856 Figure 13

- **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



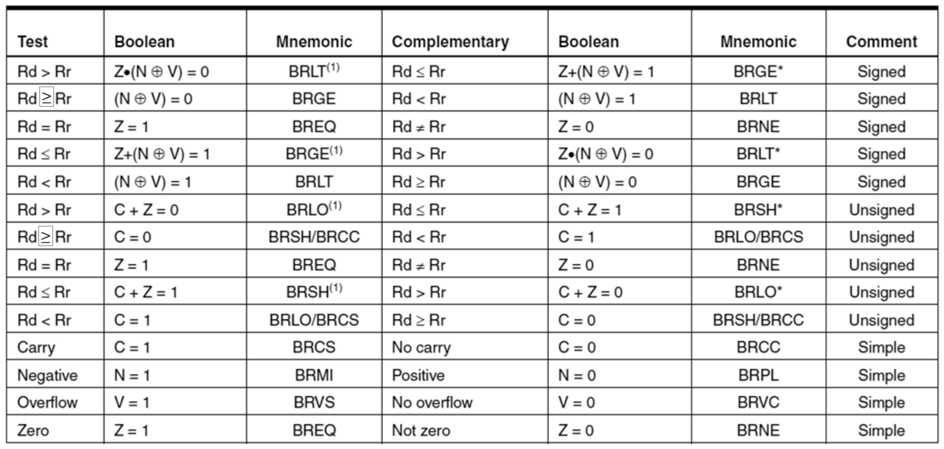
University of Florida, EEL 3744 – File 05
© Drs. Schwartz & Arroyo
28

EEL 3744 Extended Addressing Register - **XMEGA**

- EIJMP and EICALL use Extended Addressing Modes
  - >Require the use of the Extended Indirect Register (EIND), which is concatenated with the Z-register
  - >EIND holds the MSB of the 24-bit address
  - >Ex: **eijmp ;** PC(15:0) ← Z, PC(21:16) ← EIND

EEL 3744 **XMEGA** Conditional Branch Summary

See doc0856
Page 10



Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z•(N ⊕ V) = 0	BRLT(1)	Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd $\boxed{\geq}$ 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 $\boxed{\geq}$ 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

EEL 3744 **XMEGA** Instructions: Arithmetic and Logic

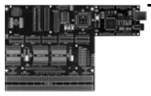
- ADD, ADC, ADIW, SUB, SUBI, SBC, SBCI, SBIW
- AND, ANDI, OR, ORI, EOR, COM, NEG
- SBR (Set Bits in Register), CBR (Clear Bits in Register)
- INC, DEC
- TEST, CLR (Clear Register), SER (Set Register)
- MUL, MULS, MULSU, FMUL, FMULS, FMULSU
- DES (Data Encryption)

Complete Instruction Summary in doc0856 Pages 11-15

EEL 3744 **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, SBIC, SBIS (Skip if bit ---)

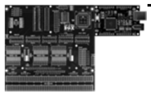
Complete Instruction Summary in doc0856 Pages 11-15



EEL 3744 **XMEGA** Instructions: Data Transfer

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

Complete Instruction Summary in doc0856 Pages 11-15



EEL 3744 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
- 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 Pages 11-15

EEL 3744 XMEGA Instructions: MCU Control

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

Complete Instruction Summary in doc0856 Pages 11-15

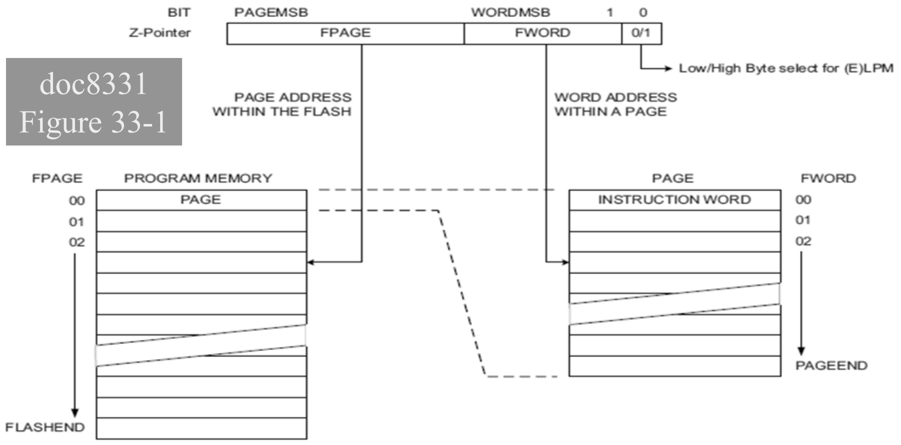
University of Florida, EEL 3744 – File **05**
© Drs. Schwartz & Arroyo

35

EEL 3744 XMEGA - Addressing the Flash

See doc8331 Sec 33.11.1.2

- Flash is **word** accessed and organized in pages
- Z-pointer is used to hold the flash memory address for read and write access. The least- significant bits address the words within a page, while the most-significant bits address the page within the flash.



doc8331 Figure 33-1

36

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`
`0x0002 0b0000 0000 0000 001_`	`0x0003`	`0x0002`
`0x0004 0b0000 0000 0000 010_`	`0x0005`	`0x0004`
…	…	…
`0x7FFF 0b0111 1111 1111 111_`	`0xFFFE`	`0xFFFF`

EEL 3744

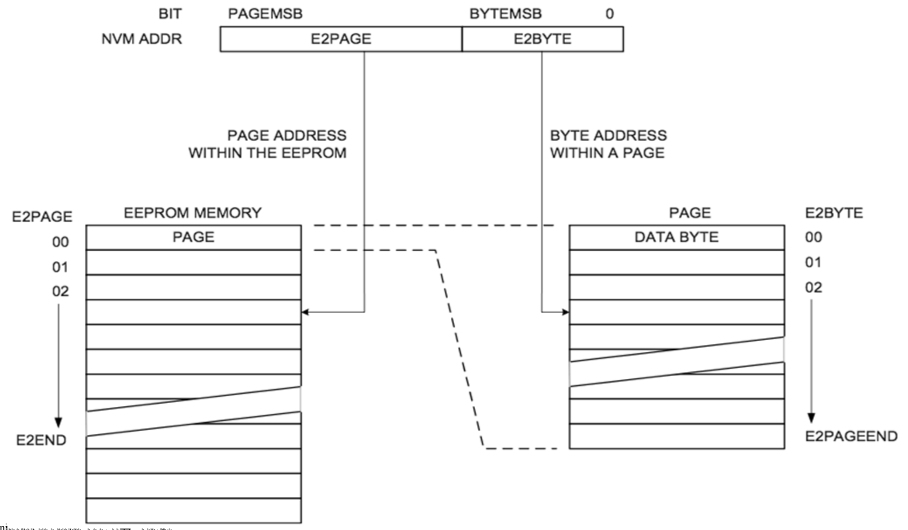
XMEGA - Addressing the EEPROM

See doc8331
Section 33-11-4

- EEPROM can be accessed through the NVM controller (I/O mapped), or it can be memory mapped into the data memory space.
- When accessing the EEPROM through the NVM controller, the NVM address (ADDR) register is used to address the EEPROM, while the NVM data (DATA) register is used to store or load EEPROM data.
- For EEPROM page programming, the ADDR register can be treated as having two sections. The least-significant bits address the bytes within a page, while the most-significant bits address the page within the EEPROM.

EEL 3744 XMEGA - Addressing the EEPROM

See doc8331 Figure 33-2



EEL 3744 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)

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

- **ret** (Return from Subroutine)
  > General format: **ret**
  > Description: PC ← STACK
  SP ← SP+2

EEL 3744

681* Instruction Set

- The instructions on the subsequent pages are a subset of the available 68HC12 instructions. The instructions can be divided up into the following 5 categories:
  - >Move Instructions
  - >Arithmetic Instructions
  - >Logic Instructions
  - >Edit Instructions
  - >Control Instructions

Instruction Bible

S&H: Chap 4

S&H: Tab 4.1

EEL 3744

Move Instructions for **681***

LDAA*	STAA**	TAB	PSHA	CLRA	XGDX
LDAB*	STAB**	TBA	PULA	CLRB	XGDY
LDD*	STD**	TAP	PSHB	CLR***	
LDX*	STX**	TPA	PULB		
LDY*	STY**	TSX	PSHX		
LDS*	STS**	TXS	PULX		
		TSY	PSHY		
		TYS	PULY		

*Memory accessed by direct, extended, indexed, or immediate addressing

***Memory accessed by extended or indexed addressing

**Memory accessed by direct, extended, or indexed addressing

EEL 3744 Arithmetic Instructions for **681***

ADDA*, ADDB*
ADCA*, ADCB*
ABA
SUBA*, SUBB*
SBCA*, SBCB*
SBA
CMPA*, CMPB*
CBA
TSTA, TSTB
TST**

INCA, INCB, INC**
DECA, DECB, DEC**
NEGA, NEGB, NEG**
ASLA, ASLB, ASL**
ASRA, ASRB, ASR**
LSLA, LSLB, LSL**
LSRA, LSRB, LSR**

Special: DAA, MUL, ABX, ABY

ADDD*
SUBD*
CPD*, CPX*, CPY*
INX, INY, INS
DEX, DEY, DES
FDIV, IDIV
ASLD, LSRD, LSLD

**Memory accessed by extended or indexed addressing

*Memory accessed by direct, extended, indexed, or immediate addressing

EEL 3744 Logic Instructions for **681***

EORA*, EORB*
ORAA*, ORAB*
ANDA*, ANDB*
BITA*, BITB*

COMA, COMB, COM**
SEC, SEI, SEV
CLC, CLI, CLV
BSET***, BCLR***

*Memory accessed by direct, extended, or indexed addressing

**Memory accessed by extended or indexed addressing

***A word in memory is specified using direct or indexed addressing, and (after a space) a pattern of bits (a mask) is specified.

EEL 3744 Edit Instructions for **681***

ASLA, ASLB, ASL*
ASRA, ASRB, ASR*

LSLA, LSLB, LSL*
LSRA, LSRB, LSR*

ROLA, ROLB, ROL*
RORA, RORB, ROR*

ASLD, LSRD, LSLD

*Memory may accessed by extended or indexed addressing

University of Florida, EEL 3744 – File **05**
© Drs. Schwartz & Arroyo

45

EEL 3744 Control Instructions for **681***

- Unconditional: JMP*, BRA**, BRN**, NOP (=skip two E-Clocks, use 1 byte)
- Conditional Simple: BEQ**, BNE**, BMI**, BPL**, BCS**, BCC**, BVS**, BVC**
- Conditional 2's Complement: BGT**, BGE**, BEQ**, BLE**, BLT**
- Conditional Unsigned: BHI**, BHS**, BEQ**, BLS**, BLO**
- Bit Conditional: BRSET***, BRCLR***
- Subroutine & Interrrupt: JSR*, BSR**, RTS, RTI, SWI, STOP, WAI

*Memory may accessed by extended or indexed addressing

**Relative addressing specifies which address to branch to.

***A word in memory is specified using direct or indexed addressing, and (after a space) a pattern of bits (a mask) is specified, and (after a space) a relative address is specified.

University of Florida, EEL 3744 – File **05**
© Drs. Schwartz & Arroyo

46

EEL 3744 Other Useful **6812** Instructions

- MOVB, MOVW, TFR, EXG
- PSHD, PSHC, PULD, PULC
- Long branches (same but start with L, e.g., LBEQ)

EEL 3744 Subroutine Control Instructions for **68HC11**

2: for Direct or Indexed X
3: for Extended or Indexed Y

- BSR (Branch to Subroutine)
  - > General format: BSR offset
  - > Addressing Mode: PC Relative ($-128 \le$ offset ≤ 127)
  - > Description: (PC) ← (PC) + 2; ((SP)) ← (PC_L); (SP) ← (SP) − 1; ((SP)) ← (PC_H); (SP) ← (SP) − 1; PC ← PC + offset
- JSR (Jump to Subroutine)
  - > General format: JSR address (or label)
  - > Addressing Mode: Direct, Extended, Indexed X, Indexed Y
  - > Description: (PC) ← (PC)+2/3; ((SP)) ← (PC_L); (SP) ← (SP) −1; ((SP)) ← (PC_H); (SP) ← (SP) − 1; PC ← addr
- RTS (Return from Subroutine)
  - > General format: RTS
  - > Addressing Mode: Inherent
  - > Description: (SP)←(SP)+1; (PC_H)←((SP)); (SP)←(SP)+1; (PC_L)←((SP))

EEL 3744 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+2
    SP ← SP–2
    PC ← PC+k+1 (constant address operand)
- **ret** (Return from Subroutine)
  - > General format: **ret**
  - > Description: PC ← STACK
    SP ← SP+2

EEL 3744

The End!