Example 2.58 Suppose the following variables are allocated in the specified order. Let `r` be a pointer that points to `l`, `q` be a pointer that points to `s`, and `p` be a pointer that points to `c`. Also, let `s` be a 16-bit `short`, `l` be a 32-bit `long`, and `c` be an 8-bit `char`. Note that in this hypothetical example, pointers are 32-bit variables meaning the processor has 32-bit addressable space.

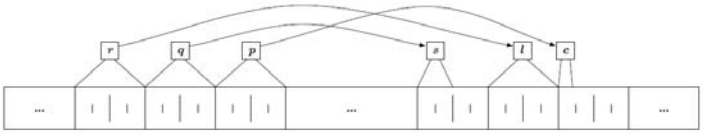

Figure 2.6: A schematic view of memory. The smallest division represents one 8-bit memory location. The height of the division line represents the byte-, short-, and long-address boundaries.

As indicated by Ex. 2.58, the amount of space reserved for any pointer is the same, no matter to what type the pointer is pointing. In the example, 32-bits are assumed for the addressable space; this is based on the architecture of the CPU. One way to make the pointer point to some variable is via the unary operator `&`, which gives the address of the label on its right-hand-side.

Example 2.59

```
p = &c;
q = &s;
r = &l;
```

Note that &p is the address of the pointer variable.

One way to access the contents of a variable using a pointer is via the unary operator *, which is called the *dereferencing* operator.

Example 2.60 At the end of these three statements, variable `c` is loaded with the value 10 via the dereference of its address in pointer `p`.

```
p = &c;
c = 0;
*p = 10;

/* now it is true that (c == 10) */
```

To declare a pointer, just add the * symbol to the left of the variable name.

Example 2.61

```
char *p;
short *q;
long *r;
```

This syntax is intended as a mnemonic. Using Ex. 2.61, the notation implies that the expression `*p` is a `char`, `*q` is a `short` and `*r` is a `long`. Note that as seen before, the space allocated to hold `p`, `q` and `r` is all the same (usually 32-bits on modern microprocessors), but what they point to is different. This matters to the compiler when pointer indexing is used.

The unary operators * and & have a very high precedence. However, the unary operators ++ and -- have the same level of operator precedence. When the compiler parses a line of source code, it resolves operators with the same precedence from right-to-left. Thus, the statement `*p++;` will have a very different result compared to `(*p)++;`. The former case will increment the address stored in `p` first, then dereference the result. The latter case will read the dereferenced address first and increment the resulting value without changing the address stored in `p`. Several examples of using pointer indexing are listed in Table 2.13, assuming that `char c = 5;`, `char *p;`, and `p = &c;`. Notice that all but the final statement are equivalent.

Table 2.13: Pointer Indexing Operations

	Before			After			
Instruction	&c = 100	101	p	&c = 100	101	p	*p
`c = *p + 1;`	5	0	100	6	0	100	6
`*p += 1;`	5	0	100	6	0	100	6
`++*p;`	5	0	100	6	0	100	6
`(*p)++;`	5	0	100	6	0	100	6
`*p++;`	5	0	100	5	0	101	0

Table 2.9: Bitwise Operators

Operator	Operation
&	AND (boolean intersection)
|	OR (boolean union)
^	XOR (boolean exclusive-or)
<<	left shift
>>	right shift
~	NOT (boolean negation, i.e., ones' complement)

Statement	c	mask	d	Embedded usefulness
`d = (c & mask);`	0x55	0x0F	0x05	Clear bits that are 0 in the mask
`d = (c | mask);`	0x55	0x0F	0x5F	Set bits that are 1 in the mask
`d = (c ^ mask);`	0x55	0x0F	0x5A	Invert bits that are 1 in the mask
`d = (c << 3);`	0x55		0xA8	Multiply by a power of 2
`d = (c >> 2);`	0x55		0x15	Divide by a power of 2
`d = ~c;`	0x55		0xAA	Invert all bits

Table 2.10: Assignment Operators

Operator	Syntax	Equivalent Operation
+=	`i += j;`	`i = (i + j);`
-=	`i -= j;`	`i = (i - j);`
*=	`i *= j;`	`i = (i * j);`
/=	`i /= j;`	`i = (i / j);`
%=	`i %= j;`	`i = (i % j);`
&=	`i &= j;`	`i = (i & j);`
|=	`i |= j;`	`i = (i | j);`
^=	`i ^= j;`	`i = (i ^ j);`
<<=	`i <<= j;`	`i = (i << j);`
>>=	`i >>= j;`	`i = (i >> j);`

Table 15-4. Waveform Generation Mode Bit Description(1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

NORMAL MODE

- The simplest AVR Timer mode of operation is the *Normal mode.* Waveform Generation Mode for Timer/Counter 1 (WGM1) bits 3:0 = 0. These bits are located in Timer/Counter Control Registers A/B (**TCCR1A** and **TCCR1B**).

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- In this mode the Timer/Counter 1 Register (TCNT1H:TCNT1L) counts up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value 0xFFFF and then restarts 0x0000.
- There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- In normal operation the Timer/Counter Overflow Flag (TOV1) bit located in the Timer/Counter1 Interrupt Flag Register (T1FR1) will be set in the same timer clock cycle as the Timer/Counter 1 Register (TCNT1H:TCNT1L) becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared.

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TIMER/COUNTER 1 PRESCALAR

- The clock input to Timer/Counter 1 (TCNT1) can be pre-scaled (divided down) by 5 preset values (1, 8, 64, 256, and 1024).

Table 13-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

- Clock Select Counter/Timer 1 (CS1) bits 2:0 are located in Timer/Counter Control Registers B.

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Normal Mode (WGM 1 bits 3:0 = 0000_2)

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 13-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

23.9 Register Description

23.9.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 23-3.

Table 23-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted.

- **Bit 4 – Res: Reserved Bit**

This bit is an unused bit in the ATmega48PA/88PA/168PA/328P, and will always read as zero.

- **Bits 3:0 – MUX3:0: Analog Channel Selection Bits**

The value of these bits selects which analog inputs are connected to the ADC.

23.9.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion.
ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the Data Registers are updated.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 23-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

23.9.3 ADCL and ADCH – The ADC Data Register

23.9.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

23.9.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

23.9.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 5:3 – Res: Reserved Bits**

These bits are reserved for future use. To ensure compatibility with future devices, these bist must be written to zero when ADCSRB is written.

- **Bit 2:0 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect.

23.9.5 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
(0x7E)	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved for future use. To ensure compatibility with future devices, these bits must be written to zero when DIDR0 is written.

- **Bit 5:0 – ADC5D..ADC0D: ADC5..0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set.