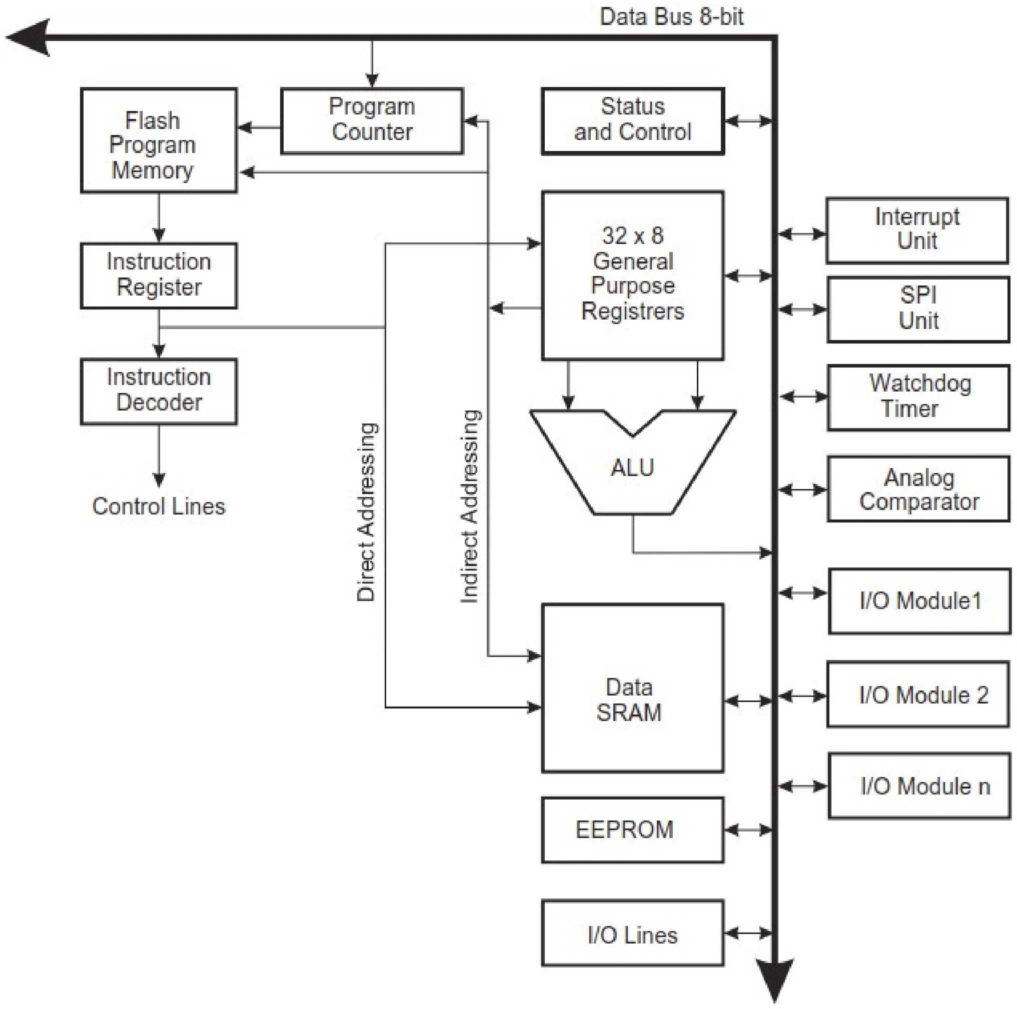
Figure 7-1. Block Diagram of the AVR Architecture



```
000000ea <setup>:
unsigned long LedTimer, Pin12Timer;
const int LedPin = 13, Pin12 = 12;

void setup() {
  // put your setup code here, to run once:
  pinMode(LedPin, OUTPUT ); // Set LED pin to output
  ea: 61 e0            ldi   r22, 0x01   ; 1
  ec: 8d e0            ldi   r24, 0x0D   ; 13
  ee: 0e 94 8f 01      call  0x31e ; 0x31e <pinMode>
  pinMode( Pin12, OUTPUT ); // Pin 12 set to output
  f2: 61 e0            ldi   r22, 0x01   ; 1
  f4: 8c e0            ldi   r24, 0x0C   ; 12
  f6: 0e 94 8f 01      call  0x31e ; 0x31e <pinMode>
  LedTimer = millis();      // Initialize timers.
  fa: 0e 94 1f 01      call  0x23e ; 0x23e <millis>
  fe: 60 93 04 01      sts   0x0104, r22
 102: 70 93 05 01      sts   0x0105, r23
 106: 80 93 06 01      sts   0x0106, r24
 10a: 90 93 07 01      sts   0x0107, r25
  Pin12Timer = millis();
 10e: 0e 94 1f 01      call  0x23e ; 0x23e <millis>
 112: 60 93 00 01      sts   0x0100, r22
 116: 70 93 01 01      sts   0x0101, r23
 11a: 80 93 02 01      sts   0x0102, r24
 11e: 90 93 03 01      sts   0x0103, r25
 122: 08 95            ret

00000124 <loop>:
}

void loop()
{
 124: 0f 93            push  r16
 126: 1f 93            push  r17
  // put your main code here, to run repeatedly:
  // Test time to see if 1 second has passed.
  if( millis() - LedTimer >= 10 )
 128: 0e 94 1f 01      call  0x23e ; 0x23e <millis>
 12c: 00 91 04 01      lds   r16, 0x0104
 130: 10 91 05 01      lds   r17, 0x0105
 134: 20 91 06 01      lds   r18, 0x0106
 138: 30 91 07 01      lds   r19, 0x0107
 13c: 60 1b            sub   r22, r16
 13e: 71 0b            sbc   r23, r17
 140: 82 0b            sbc   r24, r18
 142: 93 0b            sbc   r25, r19
 144: 6a 30            cpi   r22, 0x0A   ; 10
 146: 71 05            cpc   r23, r1
 148: 81 05            cpc   r24, r1
 14a: 91 05            cpc   r25, r1
 14c: f8 f0            brcs  .+62        ; 0x18c <loop+0x68>
  {
      // Toggle LED
      if( digitalRead( LedPin ) == HIGH )
 14e: 8d e0            ldi   r24, 0x0D   ; 13
 150: 0e 94 fe 01      call  0x3fc ; 0x3fc <digitalRead>
 154: 01 97            sbiw  r24, 0x01   ; 1
 156: 11 f4            brne  .+4         ; 0x15c <loop+0x38>
```

```
      {
          digitalWrite( LedPin, LOW );
 158: 60 e0             ldi   r22, 0x00   ; 0
 15a: 01 c0             rjmp  .+2         ; 0x15e <loop+0x3a>
      }
      else
      {
          digitalWrite( LedPin, HIGH );
 15c: 61 e0             ldi   r22, 0x01   ; 1
 15e: 8d e0             ldi   r24, 0x0D   ; 13
 160: 0e 94 c8 01       call  0x390 ; 0x390 <digitalWrite>
      } // End of LED toggle if

      LedTimer += 1000; // Update timer
 164: 80 91 04 01       lds   r24, 0x0104
 168: 90 91 05 01       lds   r25, 0x0105
 16c: a0 91 06 01       lds   r26, 0x0106
 170: b0 91 07 01       lds   r27, 0x0107
 174: 88 51             subi  r24, 0x18   ; 24
 176: 9c 4f             sbci  r25, 0xFC   ; 252
 178: af 4f             sbci  r26, 0xFF   ; 255
 17a: bf 4f             sbci  r27, 0xFF   ; 255
 17c: 80 93 04 01       sts   0x0104, r24
 180: 90 93 05 01       sts   0x0105, r25
 184: a0 93 06 01       sts   0x0106, r26
 188: b0 93 07 01       sts   0x0107, r27

      // Toggle Using function calls
      digitalWrite( Pin12, LOW );
 18c: 60 e0             ldi   r22, 0x00   ; 0
 18e: 8c e0             ldi   r24, 0x0C   ; 12
 190: 0e 94 c8 01       call  0x390 ; 0x390 <digitalWrite>
      digitalWrite( Pin12, HIGH );
 194: 61 e0             ldi   r22, 0x01   ; 1
 196: 8c e0             ldi   r24, 0x0C   ; 12
 198: 0e 94 c8 01       call  0x390 ; 0x390 <digitalWrite>

      // Toggle using Port Manipulation
      PORTB &= 0xEF;
 19c: 2c 98             cbi   0x05, 4     ; 5
      PORTB |= 0x10;
 19e: 2c 9a             sbi   0x05, 4     ; 5
      // Toggle using Port Manipulation
      // and bit functions
      bitClear(PORTB,4);
 1a0: 2c 98             cbi   0x05, 4     ; 5
      bitSet(PORTB,4);
 1a2: 2c 9a             sbi   0x05, 4     ; 5

  } // End of LedTimer if


} // End of Loop
```

```
void digitalWrite(uint8_t pin, uint8_t val)
{
 390: 0f 93           push  r16
 392: 1f 93           push  r17
 394: cf 93           push  r28
 396: df 93           push  r29
 398: 1f 92           push  r1
 39a: cd b7           in    r28, 0x3d   ; 61
 39c: de b7           in    r29, 0x3e   ; 62
      uint8_t timer = digitalPinToTimer(pin);
 39e: 28 2f           mov   r18, r24
 3a0: 30 e0           ldi   r19, 0x00   ; 0
 3a2: f9 01           movw  r30, r18
 3a4: e8 59           subi  r30, 0x98   ; 152
 3a6: ff 4f           sbci  r31, 0xFF   ; 255
 3a8: 84 91           lpm   r24, Z
      uint8_t bit = digitalPinToBitMask(pin);
 3aa: f9 01           movw  r30, r18
 3ac: e4 58           subi  r30, 0x84   ; 132
 3ae: ff 4f           sbci  r31, 0xFF   ; 255
 3b0: 14 91           lpm   r17, Z
      uint8_t port = digitalPinToPort(pin);
 3b2: f9 01           movw  r30, r18
 3b4: e0 57           subi  r30, 0x70   ; 112
 3b6: ff 4f           sbci  r31, 0xFF   ; 255
 3b8: 04 91           lpm   r16, Z
      volatile uint8_t *out;

      if (port == NOT_A_PIN) return;
 3ba: 00 23           and   r16, r16
 3bc: c9 f0           breq  .+50        ; 0x3f0 <digitalWrite+0x60>

      // If the pin that support PWM output, we need to turn it off
      // before doing a digital write.
      if (timer != NOT_ON_TIMER) turnOffPWM(timer);
 3be: 88 23           and   r24, r24
 3c0: 21 f0           breq  .+8         ; 0x3ca <digitalWrite+0x3a>
 3c2: 69 83           std   Y+1, r22    ; 0x01
 3c4: 0e 94 66 01     call  0x2cc ; 0x2cc <turnOffPWM>
 3c8: 69 81           ldd   r22, Y+1    ; 0x01

      out = portOutputRegister(port);
 3ca: e0 2f           mov   r30, r16
 3cc: f0 e0           ldi   r31, 0x00   ; 0
 3ce: ee 0f           add   r30, r30
 3d0: ff 1f           adc   r31, r31
 3d2: e2 55           subi  r30, 0x52   ; 82
 3d4: ff 4f           sbci  r31, 0xFF   ; 255
 3d6: a5 91           lpm   r26, Z+
 3d8: b4 91           lpm   r27, Z

      uint8_t oldSREG = SREG;
 3da: 9f b7           in    r25, 0x3f   ; 63
      cli();
 3dc: f8 94           cli

      if (val == LOW) {
            *out &= ~bit;
 3de: 8c 91           ld    r24, X
```

```
      out = portOutputRegister(port);

      uint8_t oldSREG = SREG;
      cli();

      if (val == LOW) {
 3e0: 61 11             cpse  r22, r1
 3e2: 03 c0             rjmp  .+6         ; 0x3ea <digitalWrite+0x5a>
            *out &= ~bit;
 3e4: 10 95             com   r17
 3e6: 81 23             and   r24, r17
 3e8: 01 c0             rjmp  .+2         ; 0x3ec <digitalWrite+0x5c>
      } else {
            *out |= bit;
 3ea: 81 2b             or    r24, r17
 3ec: 8c 93             st    X, r24
      }

      SREG = oldSREG;
 3ee: 9f bf             out   0x3f, r25   ; 63
}
 3f0: 0f 90             pop   r0
 3f2: df 91             pop   r29
 3f4: cf 91             pop   r28
 3f6: 1f 91             pop   r17
 3f8: 0f 91             pop   r16
 3fa: 08 95             ret
```