Overview

- Shift and bit-set instructions in AVR
- Sample AVR assembly programs using shift and bit-set instructions

Selected Shift and Bit-set Instructions

- Shift instructions: lsl, lsr, rol, ror, asr
- Bit-set Instructions: bset, bclr, sbi, cbi, bst, bld, sex, clx, nop, sleep, wdr, break
- Refer to AVR Instruction Set for a complete list.

Logical Shift Left

- Syntax: lsl Rd
- Operands: Rd ∈ {r0, r1, ..., r31}
- Operation: C ← Rd7, Rd7 ← Rd6, Rd6 ← Rd5, ..., Rd1 ← Rd0, Rd0 ← 0

- Flags affected: H, S, V, N, Z, C

C ← Rd7
Set if, before the shift, the MSB of Rd was set; cleared otherwise.
N ← R7
Set if MSB of the result is set; cleared otherwise.
V ← N ⊕ C
S ← N ⊕ V For signed tests.
Logical Shift Left (Cont.)

- Encoding: 0000 11dd dddd dddd
- Words: 1
- Cycles: 1
- Example: add r0, r4 ; Add r4 to r0
  lsl r0 ; Multiply r0 by 2
- Comments: This operation effectively multiplies a one-byte signed or unsigned integer by two.

Logical Shift Right

- Syntax: lsr Rd
- Operands: Rd ∈ {r0, r1, ..., r31}
- Operation: C ← Rd0, Rd0 ← Rd1, Rd1 ← Rd2, ..., Rd6 ← Rd7, Rd7 ← 0
- Flags affected: H, S, V, N, Z, C
- Encoding: 1001 010d dddd 0110
- Words: 1
- Cycles: 1
- Example: add r0, r4 ; Add r4 to r0
  lsr r0 ; Divide r0 by 2
- Comments: This instruction effectively divides an unsigned one-byte integer by two. C stores the remainder.

Rotate Left Through Carry

- Syntax: rol Rd
- Operands: Rd ∈ {r0, r1, ..., r31}
- Operation: temp ← C, C ← Rd7, Rd7 ← Rd6, Rd6 ← Rd5, ..., Rd1 ← Rd0, Rd0 ← temp

- Flag affected: H, S, V, N, Z, C
  C ← Rd7
  Set if, before the shift, the MSB of Rd was set; cleared otherwise.
  N ← R7
  Set if MSB of the result is set; cleared otherwise.
  V ← N ⊕ C
  S ← N ⊕ V For signed tests.
- Encoding: 0001 11dd dddd dddd
- Words: 1
- Cycles: 1
Rotate Left Through Carry (Cont.)

- Example: Assume a 32-bit signed or unsigned integer $x$ is stored in registers $r13:r12:r11:r10$. The following code computes $2^4 x$.

\[
\begin{align*}
\text{lsr } r10 & : \text{Shift byte 0 (least significant byte) left} \\
\text{rol } r11 & : \text{Shift byte 1 left through carry} \\
\text{rol } r12 & : \text{Shift Byte 2 left through carry} \\
\text{rol } r13 & : \text{Shift Byte 3 (most significant byte) left through carry}
\end{align*}
\]

Rotate Right Through Carry (Cont.)

- Example: Assume a 32-bit signed or unsigned integer $x$ is stored in registers $r13:r12:r11:r10$. The following code computes $x/2$.

\[
\begin{align*}
\text{asr } r13 & : \text{Shift byte 3 (most significant byte) right} \\
\text{ror } r12 & : \text{Shift byte 2 right through carry} \\
\text{ror } r11 & : \text{Shift Byte 1 right through carry} \\
\text{ror } r10 & : \text{Shift Byte 0 (least significant byte) right through carry}
\end{align*}
\]
Arithmetic Shift Right (Cont.)

• Example
  ldi r10, 10 ; r10=10
  ldi r11, -20 ; r11=-20
  add r10, r20 ; r10=-20+10
  asr r10 ; r10=(-20+10)/2

• Comments: This instruction effectively divides a signed value by two. C stores the remainder.

Bit Set in Status Register

• Syntax: bset s
• Operation: Bit s of SREG (Status Register)←1
• Operands: 0 ≤ s ≤ 7
• Flags affected
  I: 1 if s = 7; Unchanged otherwise.
  T: 1 if s = 6; Unchanged otherwise.
  H: 1 if s = 5; Unchanged otherwise.
  S: 1 if s = 4; Unchanged otherwise.
  V: 1 if s = 3; Unchanged otherwise.
  N: 1 if s = 2; Unchanged otherwise.
  Z: 1 if s = 1; Unchanged otherwise.
  C: 1 if s = 0; Unchanged otherwise.

Bit Set in Status Register (Cont.)

• Encoding: 1001 0100 0sss 1000
• Words: 1
• Cycles: 1
• Example
  bset 0 ; Set C
  bset 3 ; Set V
  bset 7 ; Enable interrupt

Bit Clear in Status Register

• Syntax: bclr s
• Operation: Bit s of SREG (Status Register)←0
• Operands: 0 ≤ s ≤ 7
• Flags affected
  I: 0 if s = 7; Unchanged otherwise.
  T: 0 if s = 6; Unchanged otherwise.
  H: 0 if s = 5; Unchanged otherwise.
  S: 0 if s = 4; Unchanged otherwise.
  V: 0 if s = 3; Unchanged otherwise.
  N: 0 if s = 2; Unchanged otherwise.
  Z: 0 if s = 1; Unchanged otherwise.
  C: 0 if s = 0; Unchanged otherwise.
Bit Clear in Status Register (Cont.)

- Encoding: 1001 0100 1sss 1000
- Words: 1
- Cycles: 1
- Example
  
  bclr 0 ; Clear C
  bclr 3 ; Clear V
  bclr 7 ; Disable interrupt

Set Bit in I/O Register

- Syntax: sbi A, b
- Operation: Bit b of the I/O register with address A←1
- Operands: 0 ≤ A ≤ 31 and 0 ≤ b ≤ 7
- Flags affected: None
- Encoding: 1001 1010 AAAA Abbb
- Words: 1
- Cycles: 2
- Example
  
  out $1E, r0 ; Write EEPROM address
  sbi $1C, 0 ; Set read bit in EECR
  in r1, $1D ; Read EEPROM data

Clear Bit in I/O Register

- Syntax: cbi A, b
- Operation: Bit s of the I/O register with address A←0
- Operands: 0 ≤ A ≤ 31 and 0 ≤ b ≤ 7
- Flags affected: None
- Encoding: 1001 1000 AAAA Abbb
- Words: 1
- Cycles: 2
- Example
  
  cbi $12, 7 ; Clear bit 7 in Port D

Set Flags

- Syntax: sex
- where x ∈ {I, T, H, S, V, N, Z, C}
- Operation: Flag x←1
- Operands: None
- Flags affected: Flag x←1
- Encoding: Depends on x.
  
  Refer to the AVR Instruction Set for details of encoding.
- Words: 1
- Cycles: 1
Set Flags (Cont.)

- Example

  sec ; Set carry flag
  adc r0, r1 ; r0=r0+r1+1
  sec
  sbc r0, r1 ; r0=r0–r1–1
  sen ; Set negative flag
  sei ; Enable interrupt
  sev ; Set overflow flag
  sez ; Set zero flag
  ses ; Set sign flag

Clear Flags

- Syntax: clx

  where \( x \in \{I, T, H, S, V, N, Z, C\} \)

- Operation: Flag \( x \leftarrow 0 \)

- Operands: None

- Flags affected: Flag \( x \leftarrow 0 \)

- Encoding: Depends on \( x \).

  Refer to the AVR Instruction Set for details of encoding.

- Words: 1

- Cycles: 1

Clear Flags

- Example

  clc ; Clear carry flag
  cln ; Clear negative flag
  cli ; Disable interrupt
  clv ; Clear overflow flag
  clz ; Clear zero flag
  cls ; Clear sign flag

No Operation

- Syntax: nop

- Operation: No

- Operands: None

- Flags affected: None

- Encoding: 0000 0000 0000 0000

- Words: 1

- Cycles: 1

- Example

  clr r16 ; Clear r16
  ser r17 ; r17=0xff
  out $18, r16 ; Write zeros to Port B
  nop ; Wait (do nothing)
  out $18, r17 ; Write ones to Port B
Sleep

• Syntax: sleep

• Operation: Sets the circuit in sleep mode defined by the MCU control register. When an interrupt wakes the MCU from the sleep mode, the instructions following the sleep instruction will be executed.

• Operands: None

• Flags affected: None

• Encoding: 1001 0101 1000 1000

• Words: 1

• Cycles: 1

• Example

```
ldi r16, (1<<SE) ; Enable sleep mode
out MCUCR, r16
sleep ; Put MCU in sleep mode
```

Watchdog Reset

• Syntax: wdr

• Operation: Resets the Watchdog Timer. This instruction must be executed within a limited time given by the WD prescaler. See the Watchdog Timer hardware specification.

• Operands: None

• Flags affected: None

• Encoding: 1001 0101 1010 1000

• Words: 1

• Cycles: 1

• Example

```
wdr ; Reset watchdog timer
```

Break

• Syntax: break

• Operation: The break instruction is used by the On-Chip Debug system, and is normally not used in the application software. When the BREAK instruction is executed, the AVR CPU is set in the Stopped Mode. This gives the On-Chip Debugger access to internal resources.

  If any lock bits are set, or either the JTAGEN or OCDEN fuses are unprogrammed, the CPU will treat the break instruction as a nop and will not enter the Stopped Mode.

• Operands: None

• Flags affected: None

• Encoding: 1001 0101 1001 1000

• Words: 1

• Cycles: 1

• Example

```
break ; stop here
```