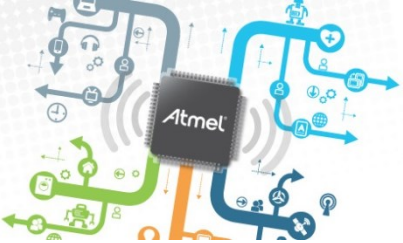
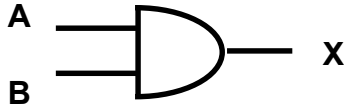
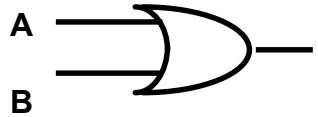
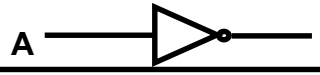
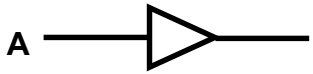
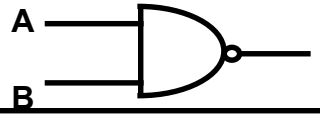
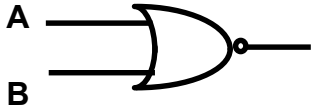
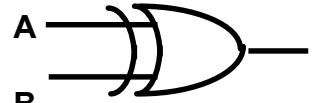
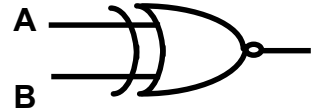
1DT301, Computer Technology

Tuesday, October 16, 2019

- Questions?
- Exam examples



Digital gates

Name	Symbol	Function	Truth Table
AND		X = A • B or X = AB	A B X 0 0 0 0 1 0 1 0 0 1 1 1
OR		X = A + B	A B X 0 0 0 0 1 1 1 0 1 1 1 1
I		X = A’	A X 0 1 1 0
Buffer		X = A	A X 0 0 1 1
NAND		X = (AB)’	A B X 0 0 1 0 1 1 1 0 1 1 1 0
NOR		X = (A + B)’	A B X 0 0 1 0 1 0 1 0 0 1 1 0
XOR Exclusive OR		X = A ⊕ B or X = A’B + AB’	A B X 0 0 0 0 1 1 1 0 1 1 1 0
XNOR Exclusive NOR or Equivalence		X = (A ⊕ B)’ or X = A’B’+ AB	A B X 0 0 1 0 1 0 1 0 0 1 1 1

INC – Increment

Description:

Adds one -1- to the contents of register Rd and places the result in the destination register Rd.

The C Flag in SREG is not affected by the operation, thus allowing the INC instruction to be used on a loop counter in multiple-precision computations.

When operating on unsigned numbers, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

Operation:

(i) Rd ← Rd + 1

	Syntax:	Operands:	Program Counter:
(i)	INC Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0011

Status Register and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	–

S: $N \oplus V$
For signed tests.

V: $R7 \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was $7F before the operation.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; Cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
        clr     r22         ; clear r22
loop:   inc     r22         ; increment r22
        ...
        cpi     r22,$4F     ; Compare r22 to $4f
        brne    loop        ; Branch if not equal
        nop                 ; Continue (do nothing)
```

MUL – Multiply Unsigned

Description:

This instruction performs 8-bit × 8-bit → 16-bit unsigned multiplication.

Rd		Rr		R1	R0
Multiplicand	×	Multiplier	→	Product High	Product Low
8		8		16	

The multiplicand Rd and the multiplier Rr are two registers containing unsigned numbers. The 16-bit unsigned product is placed in R1 (high byte) and R0 (low byte). Note that if the multiplicand or the multiplier is selected from R0 or R1 the result will overwrite those after multiplication.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 ← Rd × Rr (unsigned ← unsigned × unsigned)

	Syntax:	Operands:	Program Counter:
(i)	MUL Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

1001	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	⇔	⇔

C: R15
Set if bit 15 of the result is set; cleared otherwise.

Z: $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

```
mul   r5,r4     ; Multiply unsigned r5 and r4
movw  r4,r0     ; Copy result back in r5:r4
```

Words: 1 (2 bytes)
Cycles: 2

MULS – Multiply Signed

Description:
This instruction performs 8-bit × 8-bit → 16-bit signed multiplication.

Rd		Rr		R1	R0
Multiplicand	×	Multiplier	→	Product High	Product Low
8		8		16	

The multiplicand Rd and the multiplier Rr are two registers containing signed numbers. The 16-bit signed product is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 ← Rd × Rr (signed ← signed × signed)

	Syntax:	Operands:	Program Counter:
(i)	MULS Rd,Rr	$16 \le d \le 31$, $16 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0000	0010	dddd	rrrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	⇔	⇔

C: R15
Set if bit 15 of the result is set; cleared otherwise.

Z: $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

```
muls  r21,r20   ; Multiply signed r21 and r20
movw  r20,r0    ; Copy result back in r21:r20
```

Words: 1 (2 bytes)
Cycles: 2

MULSU – Multiply Signed with Unsigned

Description:
This instruction performs 8-bit × 8-bit → 16-bit multiplication of a signed and an unsigned number.

Rd		Rr		R1	R0
Multiplicand	×	Multiplier	→	Product High	Product Low
8		8		16	

The multiplicand Rd and the multiplier Rr are two registers. The multiplicand Rd is a signed number, and the multiplier Rr is unsigned. The 16-bit signed product is placed in R1 (high byte) and R0 (low byte).

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) R1:R0 ← Rd × Rr (signed ← signed × unsigned)

	Syntax:	Operands:	Program Counter:
(i)	MULSU Rd,Rr	$16 \le d \le 23$, $16 \le r \le 23$	PC ← PC + 1

16-bit Opcode:

0000	0011	0ddd	0rrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	⇔	⇔

C: R15
Set if bit 15 of the result is set; cleared otherwise.

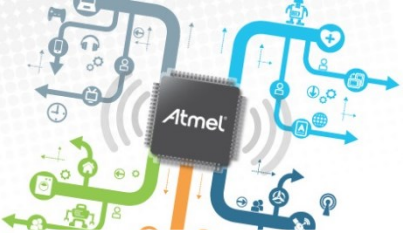
Z: $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $0000; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

```
;*****************************************************************************
;* DESCRIPTION
;*Signed multiply of two 16-bit numbers with 32-bit result.
;* USAGE
;*r19:r18:r17:r16 = r23:r22 * r21:r20
;*****************************************************************************
muls16x16_32:
  clrr2
  mulsr23, r21; (signed)ah * (signed)bh
```

- unsigned 8 bit: 0 - 255
- unsigned 16 bit: 0 – 65535
- signed 8 bit: -128 - +127
- signed 16 bit: -32768 - + 32767
- signed 8 bit x signed 8 bit = signed 16 bit
- unsigned 8 bit x signed 8 bit = signed 16 bit
- unsigned 8 bit x unsigned 8 bit = unsigned 16 bit

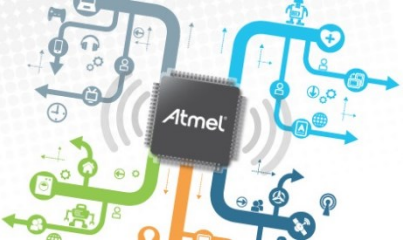


1.

a) Convert the hexadecimal number $\mathbf{12B0_{16}}$ and the decimal number $\mathbf{4784_{10}}$ to binary numbers and add the two binary numbers together. Give answer in hexadecimal! Show the calculations. **2p**

b) Write the octal number $\mathbf{4005210046_{8}}$ in hexadecimal form! **1p**

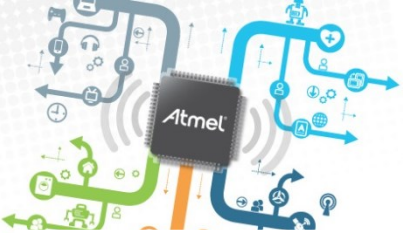
c) Write the decimal number -6 as a hexadecimal value in two's-complement form, 16 bits. **2p**



4784 to binary:

Division:	Quote:	Rest:		
4784/2	2392	0	lsb	= least significant bit
2392/2	1196	0		
1196/2	598	0	4 bits	
598/2	299	0		
299/2	149	1		
149/2	74	1	4 bits	
74/2	37	0		
37/2	18	1		
18/2	9	0		
9/2	4	1	4 bits	
4/2	2	0		
2/2	1	0		
1/2	0	1	msb	= most significant bit

The binary number is: 1 0010 1011 0000 = 0x12B0



1.

a) Convert the hexadecimal number **$12B0_{16}$** and the decimal number **4784_{10}** to binary numbers and add the two binary numbers together. Give answer in hexadecimal! Show the calculations. **2p**

b) Write the octal number **4005210046_8** in hexadecimal form! **1p**

c) Write the decimal number -6 as a hexadecimal value in two's-complement form, 16 bits. **2p**

a) Convert the hexadecimal number **$12B0_{16}$** and the decimal number **4784_{10}** to binary numbers and add the two binary numbers together. Give answer in hexadecimal! Show the calculations. **2p**

Answer: $0001\ 0010\ 1011\ 0000_2 + 0001\ 0010\ 1011\ 0000_2 = 0010\ 0101\ 0110\ 0000_2 = 2560_{16}$

b) Write the octal number **4005210046_8** in hexadecimal form! **1p**

Answer: $4005210046_8 = 100\ 000\ 000\ 101\ 010\ 001\ 000\ 000\ 100\ 110_2 = 20151026_{16}$

c) Write the decimal number -6 as a hexadecimal value in two's-complement form, 16 bits. **2p**

Answer: $+6_{10} = 0000\ 0000\ 0000\ 0110_2$ (16 bits)

1-complement = $1111\ 1111\ 1111\ 1001_2$

2-complement = 1-complement +1 = $1111\ 1111\ 1111\ 1010_2 = FFFA_{16}$

2. Machine code

Below is part of a progran
most significant byte first,
Example: The instruction
On lines B1 – B9 the ass

a) Recreate the assembler c
Instruction Set Manual.

b) Make a flowchart of the p

```
+000000B1:   9731
+000000B2:   F7F1
+000000B3:   9508

+000000B4:   27FF
+000000B5:   E1EE
+000000B6:   CFFA

+000000B7:   E0F3
+000000B8:   EEE8
+000000B9:   940C00B1
```

SBIW – Subtract Immediate from Word

Description:

Subtracts an immediate value (0-63) from a register pair and places the result in the register pair. This ins
on the upper four register pairs, and is well suited for operations on the Pointer Registers.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) Rd+1:Rd ← Rd+1:Rd - K

	Syntax:	Operands:	Program Counter:
(i)	SBIW Rd+1:Rd,K	d ∈ {24,26,28,30}, 0 ≤ K ≤ 63	PC ← PC + 1

16-bit Opcode:

1001	0111	KKdd	KKKK
1001	0111	0011	0001

K=? K=00 0001
D=? d=11 = 3
Rd=R30
Rd+1: Rd=R31:R30

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

sbiw r31:r30, 1

S: N ⊕ V, For signed tests.

V: Rdh7 •$\overline{R15}$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R15
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R15}$• $\overline{R14}$ •$\overline{R13}$ •$\overline{R12}$ •$\overline{R11}$• $\overline{R10}$• $\overline{R9}$• $\overline{R8}$• $\overline{R7}$• $\overline{R6}$ •$\overline{R5}$• $\overline{R4}$• $\overline{R3}$ •$\overline{R2}$• $\overline{R1}$• $\overline{R0}$
Set if the result is $0000; cleared otherwise.

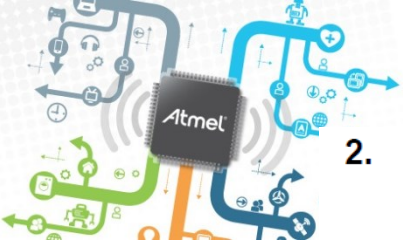
C: R15• $\overline{Rdh7}$
Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
sbiw    r25:r24,1    ; Subtract 1 from r25:r24
sbiw    YH:YL,63     ; Subtract 63 from the Y-pointer(r29:r28)
```

Words: 1 (2 bytes)
Cycles: 2



2. Machine code

Below is part of a pr
most significant byte
Example: The instru
On lines B1 – B9 the

a) Recreate the assem
Instruction Set Manu

b) Make a flowchart of

```
+000000B1:    9731
+000000B2:    F7F1
+000000B3:    9508
+000000B4:    27FF
+000000B5:    E1EE
+000000B6:    CFFA
+000000B7:    E0F3
+000000B8:    EEE8
+000000B9:    940C00B1
```

BRNE – Branch if Not Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is cleared. If the
cuted immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only
signed binary number represented in Rd was not equal to the unsigned or signed binary number repre
instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). The parameter
PC and is represented in two's complement form. (Equivalent to instruction BRBC 1,k).

Operation:

(i) If Rd ≠ Rr (Z = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRNE k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k001
1111	0111	1111	0001

K=?

K=11 1111 0 =
111 1110

K = -2

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
        eor     r27,r27   ; Clear r27
loop:   inc     r27       ; Increase r27
        ...
        cpi     r27,5     ; Compare r27 to 5
        brne    loop      ; Branch if r27<>5
        nop               ; Loop exit (do nothing)
```

PC = PC + k + 1=
PC-1

brne pc-1

Words: 1 (2 bytes)
Cycles: 1 if condition is false
2 if condition is true

2. Machine code

Below is part of a program, cut out from the disassembler. The machine code is printed with the most significant byte first, ie as it is described in the Instruction Set Manual.
Example: The instruction RET, the machine code is 9508.
On lines B1 – B9 the assembler code is removed.

a) Recreate the assembler code by interpreting machine code. Use the enclosed examples from the Instruction Set Manual.

b) Make a flowchart of the program and explain what the code is doing.

```
+000000B1:   9731

+000000B2:   F7F1

+000000B3:   9508        RET                    Subroutine return

+000000B4:   27FF

+000000B5:   E1EE

+000000B6:   CFFA

+000000B7:   E0F3

+000000B8:   EEE8

+000000B9:   940C00B1
```

2. Machine code

Below is part of a program, cut ou
most significant byte first, ie as it i
Example: The instruction RET, th
On lines B1 – B9 the assembler c

a) Recreate the assembler code by i
Instruction Set Manual.

b) Make a flowchart of the progra

```
+000000B1:    9731

+000000B2:    F7F1

+000000B3:    9508          RET

+000000B4:    27FF

+000000B5:    E1EE

+000000B6:    CFFA

+000000B7:    E0F3

+000000B8:    EEE8

+000000B9:    940C00B1
```

CLR – Clear Register

Description:

Clears a register. This instruction performs an Exclusive OR between a register and itself. This will clear register.

Operation:

(i) Rd ← Rd ⊕ Rd

	Syntax:	Operands:	Program Counter:
(i)	CLR Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode: (see EOR Rd,Rd)

0010	01dd	dddd	dddd

d = 11 1111 1111 = ?

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	0	0	0	1	–

S: 0
Cleared

V: 0
Cleared

N: 0
Cleared

Z: 1
Set

R (Result) equals Rd after the operation.

Example:

```
        clr   r18       ; clear r18
loop:   inc   r18       ; increase r18
        ...
        cpi   r18,$50   ; Compare r18 to $50
        brne  loop
```

Words: 1 (2 bytes)
Cycles: 1

2. Machine code

Below is part of a program, (
most significant byte first, ie
Example: The instruction RE
On lines B1 – B9 the assem

a) Recreate the assembler coc
Instruction Set Manual.

b) Make a flowchart of the prog

```
+000000B1:    9731

+000000B2:    F7F1

+000000B3:    9508

+000000B4:    27FF

+000000B5:    E1EE

+000000B6:    CFFA

+000000B7:    E0F3

+000000B8:    EEE8

+000000B9:    940C00B1
```

EOR – Exclusive OR

Description:

Performs the logical EOR between the contents of register Rd and register Rr and places th
ter Rd.

Operation:

(i) Rd ← Rd ⊕ Rr

	Syntax:	Operands:	Program Counter:
(i)	EOR Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0010	01rd	dddd	rrrr

d = 1 1111 d = 31
r = 1 1111 r = 31

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

S: N ⊕ V, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

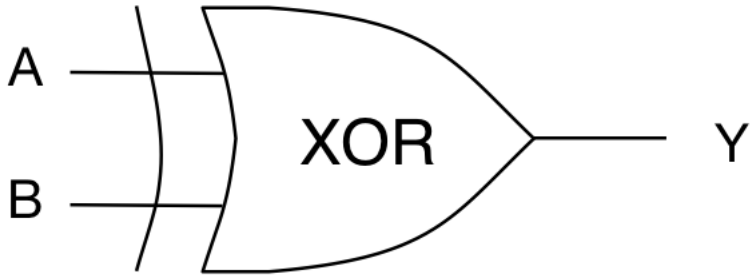
Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

R (Result) equals Rd after the operation.

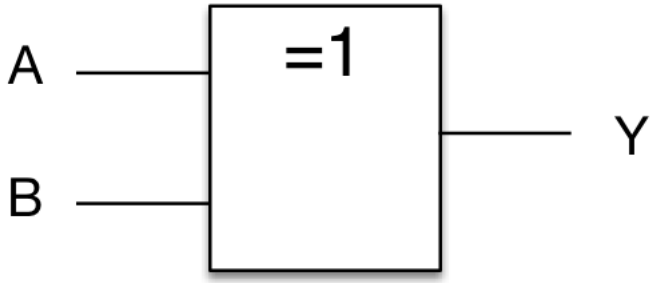
Example:

```
eor     r4,r4     ; Clear r4
```

Exclusive OR



A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0



1010 0101	1010 0101
1111 0000	1010 0101
0101 0101	0000 0000

xor r16, r16 ? = clr r16

2. Machine code

Below is part of a program, cut o
most significant byte first, ie as it i
Example: The instruction RET, th
On lines B1 – B9 the assembler c

a) Recreate the assembler code by i
Instruction Set Manual.

b) Make a flowchart of the progra

```
+000000B1:    9731

+000000B2:    F7F1

+000000B3:    9508          RET

+000000B4:    27FF

+000000B5:    E1EE

+000000B6:    CFFA

+000000B7:    E0F3

+000000B8:    EEE8

+000000B9:    940C00B1
```

CLR – Clear Register

Description:

Clears a register. This instruction performs an Exclusive OR between a register and itself. This will clear register.

Operation:

(i) Rd ← Rd ⊕ Rd

	Syntax:	Operands:	Program Counter:
(i)	CLR Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode: (see EOR Rd,Rd)

0010	01dd	dddd	dddd

d = 11 1111 1111 = 31

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	0	0	0	1	–

S: 0
Cleared

V: 0
Cleared

N: 0
Cleared

Z: 1
Set

R (Result) equals Rd after the operation.

Example:

```
        clr   r18       ; clear r18
loop:   inc   r18       ; increase r18
        ...
        cpi   r18,$50   ; Compare r18 to $50
        brne  loop
```

Words: 1 (2 bytes)
Cycles: 1