# CS/ECE 252 Exam 3 Review

11AM section 2015 November 20

### Before we get started

Homework 7 due at beginning of lecture

Exam 3 on Monday, November 23rd, during class

Homework 8 will be released on/by Wednesday, November 25 HW8 due on Friday, December 11th, at beginning of class

Quote of the day:

"Success consists of going from failure to failure without loss of enthusiasm" -- Sir Winston Churchill 1874-1965

Today: Exam 3 Review

Instruction	Description	Op 1	Op 2	Effect on operands	Effect on specials	Example
Load/Store instr	ructions	-			-	
ldi	Load immediate	reg 16-31	8-bit value	op1 = op2	incPC; sreg: none	ldi r16, 49
mov	Move	reg	reg	op1 = op2	incPC; sreg: none	mov r1, r2
ld	Load from RAM	reg	x	op1 = RAM[r26 + 256*r27]	incPC; sreg: none	ld r1, X
st	Store in RAM	x	reg	RAM[r26 + 256*r27] = op2	incPC: sreg: none	st X. r1
Computation Inst	ructions					
add	Add	eaa	nag	on1 = on1 + on2	incDC: sear: N7C	add c1 c2
aub.	Coldenate			opi = opi + opi	Andre, areg. MZC	auk a1 a2
Sub	Subchacic	reg	reg	obi = obi - obs	INCPC; SPEg: NZC	SUD P1, P2
inc	Increment register	reg	none	op1 = op1 + 1	incPC; sreg: NZ	inc rl
dec	Decrement register	reg	none	op1 = op1 - 1	incPC; sreg: NZ	dec r1
and	And	reg	reg	op1 = op1 & op2	incPC; sreg: NZ	and r1, r2
or	On	reg	reg	op1 = op1   op2	incPC: sreg: NZ	or r1. r2
eor	Exclusive-or	reg	reg	on1 = on1 ^ on2	incDC: spag: N7	eor r1 r2
	Complement, Net	1.46		opi - opi opi	inore, sreg, NZ	cor ri, ri
con	comprement, woc	1.65	none	opi = ~opi	INCPC, STEE, NZ	CON 11
neg	Negate	reg	none	ob1 = -ob1	incPC; sreg: NZC	neg ri
asr	Arithmetic shift right	reg	none	op1 = op1 >> 1	incPC; sreg: NZC	asr rl
				MSB unmodified		
cp	Compare two registers	reg	neg	none	incPC: sreg: NZC*	cp r1, r2
subi	Subtract immediate	reg 16-31	8-bit value	on1 = on1 - on2	incPC: speg: N7C	subi r16, 1
ands	And immediate	neg 15-31	8-bit value	on1 = on1 & on7	incpC: sear: N7	andi n16 1
undi	On immediate	16 16 21	o bit value	opi = opi a opi	Andrey areg. NZ	und1 110, 1
011	Or inmediate	reg 10-51	8-DIC Maine	opi = opi   opz	incrc; sreg: NZ	or1 r10, 1
adc	Add with carry	reg	reg	op1 = op1 + op2 + C	incert; sreg: NZC	add FI, FZ
sbc	Subtract with carry	reg	reg	op1 = op1 - op2 - C	incPC; sreg: NZC	sbc r1, r2
Input/Output						
in	Read I/O register	reg	value 0-63	Read op2 into op1	incPC; sreg: none	in r1, 16
out	Write to I/O register	value 0-63	reg	Write op2 to op1	incPC; sreg: none	out 18, r1
Control-Flow				•	•	
nimo	Relative jump	Value: -2048 to	none	none	nc=pc+on1:	cino 4
1.346	Here ere Janb	2047			incDC: spar: popa	nimo -2
		2047			incre, sreg, none	runp -s
breq	Branch 1† equal	Value: -64 to +63	none	none	It Z set, pc=pc+op1;	preg 2
					incPC; sreg: none	breg -3
brne	Branch if not equal	Value: -64 to +63	none	none	If Z clear, pc=pc+op1;	brne 2
					incPC; sreg: none	brne -3
brsh	Branch if same or higher	Value: -64 to +63	none	none	If C clear, nc=pc+on1:	brsh 2
or sit	pronon 21 June of higher	101001 04 00 100	none	none	incDC: coat; pope	bach -2
1.1.1	Described (C. Jacobs	11.1	2000	2000 C	There, sieg, none	b-3- 2
DLTO	Branch if lower	Value: -64 to +63	none	none	If C set, pc=pc+op1;	DLTO S
12220	10 MIN 10 MIN 10 MIN	17 JF			incPC; sreg: none	brio -3
rcall	Call relative address	Value: -2048 to	none	push pc+1	pc=pc+op1;	rcall -10
		2047			incPC; sreg: none	
ret	Return from subroutine	none	none		non no: speg: none	ret
					F - F - F - F B	
Stack						
nush	Push register to stack	rea	none	RAM[sn] = on1	incPC: sceg: none	oush_r1
pro arr	rush regarder to state	1.98	none	cp = cp = 1	andre, areg. none	pushiriz
	Den namiskan off skeek			sp _ sp - 1	in offer search search	ana a1
pop	POP Tegister off stack	1.68	none	sp = sp + 1	incre, sreg. none	pop int
			L	opi = KAM[sp]		
Setting sreg			Setting sneg for cp	·		
N: set if result	: is negative		if (op1 > op2) set N	else clear N	PIND: IO reg 16	
Z: set if result	: is 0		if (op1 == op2) set Z	else clear Z	DDRD: IO reg 17	
add, addi, adc	C: set if op1 + op2 > 1	255	if (op1 < op2) set C	else clear C	PORTD: IO reg 18	
sub subi	C: set if onl < on2 (u	signed)			SPL: TO peg 61	
nee asp	C: set if onl was odd				SPH: TO peg 62	
she	Cr set if only C c on	(upsigned)			billi 10 reg oz	
300	e. see al opa i e e op	c (augrEuca)				
Are Dissetting	Face and a			TSA Connections		
Ash Directive	Label serve			100 Operacions	141 - 36 1-0(1-0-1)	
Tapers	raper_name:			108	Idi nzo, ios(iabei_name)	
byte	.byte(label_name) 0,1,1,2	,3,5,8		h18	ldi r2/, hi8(label_name)	
string	.string(label_name) "hell	.o world"				
Program layout						
Prog memory		if $(x < y)$	if (x == y)		write reg 2 to output	
ldi r31, 91		foo	x in r1.y in r2	1	ldi r31, 255	
out 62, r31		else	cn r1, r2	1	out 17, c31	
145 -21 126		han	boar foo code	1	out 18 n3	
101 101, 100	- Sector	coll contract	preg TOO_CODE	4	out 10, 12	
ouc of, rat -	<ul> <li>set sp</li> </ul>	rescor program	10.1	4		
rjmp prog_beg	jump to program start	and the second of the second	1† (X ==10)	1		
code for functio	ns	;x in r1,y in r2	;x in r1,y in r2	1		
code for functio	ins	cp r1, r2	ldi r31, 10	1		
code for functio	ns	brlo foo_code	cp r1, r31	1		
code for functio	ins	bar line 1	breg foo code	1		
code for functio	0.0	bar line 2		1		
code for functio	115	ban line 2	14 (m. x. m)	4		
program_peg:		Dar line 3	$r(x \ge y)$	1		
code for main pr	og.	rjmp merge_point	;x in r1,y in r2	1		
code for main pr	og.	foo_code:	cp r1, r2	1		
code for main pr	og.	foo line 1	brsh foo_code	1		
code for main pr	og.	foo line 2		-		
code for main pr		foo line 3	1			
the for moth bi		manma noint:	1			
		merge_point:	1			
		rest of prog.	1			

#### Cheatsheets 1-2: Chapters 5-6

Instruction	Description	туре	Elicoutilă	схащрте	
Load/Store I	nstructions				
ldi	load immediate	4-bit reg, 8-bit imm	1110_IIII_RRRR_IIII	ldi r16, 45 = 1110_45[7:4]_r16_45[3:0]	= 1110_0010_0000_110
mov	move; copy register	5-bit reg, 5-bit reg	001011_S_RRRRR_SSSS	<pre>mov r1, r2 = 001011_r2[4]_r1_r2[3:0]</pre>	= 001011_0_00001_001
ld	load from RAM	5-bit reg	1001000_RRRRR_1100	ld r1, X = 1001000_r1_1100	= 1001000_00001_1100
st	store to RAM	5-bit reg	1001001_RRRRR_1100	st X, r1 = 1001001_r1_1100	= 1001001_00001_110
Computation	Instructions				
add	add without carry	5-hit peg 5-hit peg	AGAIL S PRABE SSSS	add r1 r2 - 000011 r2[4] r1 r2[3-0]	- 000011 0 00001 00
sub	subtract without carry	5-hit reg. 5-hit reg	000110 S RRRR SSSS	sub c1, c2 = $000110$ c2[4] c1 c2[3:0]	= 000110 0 00001 00
inc	increment	5-hit reg	1001010 RRRRR 0011	inc $c1 = 1001010 c1 0011$	= 1001010 00001 001
dec	decrement	5-hit peg	1001010 RRRRR 1010	dec c1 = 1001010 c1 1010	= 1001010_00001_001
and	logical AND	5-bit peg 5-bit peg	AA1AAA C PPPPP CCCC	and $n1 = n2 = 0.000010 [11] [010 = 0.00000000000000000000000000000000$	- 001000 0 00001 001
anu	logical OP	S-bit reg, S-bit reg	GOIGO S RANK SSSS	and $11, 12 = 001000 [2[4] [1 [2[5:0]]]$	- 001000_0_00001_00
00	logical avelusius OD	E hit and E hit and	001010_5_MMMM_5555	$p_{1} = p_{1} = p_{2} = p_{1} = p_{1$	- 001001 0 00001 00
com	logical complements NOT	E hit and	1001001_3_MMMM_3333	con = 1 = 1001001 [2[4] [1 [2[3,0]]	- 1001001_0_00001_00
Con	rogical complement; NOT	E hit nog	1001010_NNNN_0000	$r_{11} = 1001010 [1] 0000$	- 1001010_00001_000
neg	negate	S-bit reg	1001010_NKKKK_0001	neg r1 = 1001010_r1_0001	= 1001010_00001_000
asr	arithmetic shift right	S-Dit reg	1001010_RRRRR_0101		= 1001010_00001_010.
ср	compare	5-Dit reg, 5-Dit reg	000101_5_RRRRR_5555	cp  r1,  r2 = 000101 r2[4] r1 r2[3:0]	= 000101_0_00001_00.
subi	subtract immediate	4-bit reg, 8-bit imm	0101_1111_RRRR_1111	$sub1 r16, 45 = 0101_{45}[7:4]_r16_{45}[3:0]$	= 0101_0010_0000_11
andı	logical AND immediate	4-bit reg, 8-bit imm	0111_IIII_RRRR_IIII	and $r_{16}$ , $45 = 0111_{45}[7:4]_{r_{16}}[45[3:0]$	= 0111_0010_0000_11
ori	logical OR immediate	4-bit reg, 8-bit imm	0110_IIII_RRRR_IIII	ori r16, $45 = 0110_45[7:4]_r16_45[3:0]$	= <b>0110</b> _0010_0000_11
cpi	compare with immediate	4-bit reg, 8-bit imm	0011_IIII_RRRR_IIII	cpi r17, 45 = 0011_45[7:4]_r17_45[3:0]	= <b>0011</b> _0010_ <b>0001</b> _11
adc	add with carry	5-bit reg, 5-bit reg	000111_S_RRRRR_SSSS	adc r1, $r2 = 000111 r2[4] r1 r2[3:0]$	= 000111_0_00001_00
sbc	subtract with carry	5-bit reg, 5-bit reg	000010_S_RRRRR_SSSS	<pre>sbc r1, r2 = 000010_r2[4]_r1_r2[3:0]</pre>	= 000010_0_00001_00
Input/Output	Instructions		and the state of the		
in	load from I/O register	5-bit reg, I/O reg	10110 AA RRRRR AAAA	in r1, 16 = 10110 I016[5:4] r1 I016[3:0]	= 10110 01 00001 00
out	store to I/O register	5-bit reg, I/O reg	10111_AA_RRRRR_AAAA	out 18, r1 = 10111_I018[5:4]_r1_I018[3:0]	= 10111_01_00001_00
Control Elau	Instructions				
conci o1-Fiow	polativo jumo	12 bit imm	1100 TTTTTTTTTTTT	nim 4 - 1100 1	- 1100 000000000100
hceo	hearch if equal	7-hit imm	111100 TITTITI 001	hpen 2 = 111100 2 001	- 111100 000000010
hone	branch if not equal	7-bit imm	111101 TITTTT 001	hone 2 - 111101 2 001	- 111101 0000010_00
hosh	branch if same/higher	7-bit imm	111101_111111_001	high 2 - 111101 2 000	- 111101_0000010_00
halo	branch if lower	7-bit imm	111100_1111111_000	halo 2 = 111101_2_000	- 111101_0000010_00
ncall	palative call subporting	12-hit imm	1100_1111111_000	ncall -22 = 1100 2 000	- 1100_0000010_00
nat	neture from subsouting	othon	1001_1111111111111	- 1001 0101 0000 1000	- 1001_11111101001
i et	recurn from subroucine	ouren	1001_0101_0000_1000	150 - 1001 0101 0000 1000	- 1001_0101_0000_100
Stack Instru	ctions		-		
push	push register to stack	5-bit reg	1001001_RRRRR_1111	push r1 = 1001001_r1_1111	= 1001001_00001_111
рор	pop register off stack	5-bit reg	1001000_RRRRR_1111	pop r1 = 1001000_r1_1111	= 1001000_00001_111:
Format Legen	d:	Туре	Format	1	
underscores	are used for readability	5-bit reg	CCCCCCC RRRRR CCCC	1	
C = opcode		5-bit reg, 5-bit reg	CCCCCC S RRRRR SSSS	1	
R = first re	gister	5-hit reg. T/O reg	CCCCC AA BRBRB AAAA	1	
S = second n	egister	4-bit reg. 8-bit imm	CCCC TITL RRRR TITT	1	
A = T/O pegi	ster	7-bit imm	CCCCCC TITITIT CCC	1	
The state of the s		10 624 2	CCCC ITITITITI	4	

CCCC CCCC CCCC CCCC

other



to 16 bits



### Cheatsheet 4: Chapter 7, Microarchitecture



### Exam 3 Topics - Chapter 6

Encoding Principles - section6.2, lecture18\_slide18

Encoding - section6.3.2, HW6Q13; lecture18

Decoding - section7.2.1.1, section7.3.1through7.3.7, HW6Q14; lecture18

## Exam 3 Topics - Chapter 7

Microarchitecture - section7.2, lecture19slide3-5

State Machines - section7.2through7.2.1.1, HW7Q1, HW7Q2, HW7Q11, HW7Q12, lecture19slide8-15, lecture22slide4-11

Microarchitecture execution stages - lecture19\_slide10, section7.2through7.2.1.1, HW7Q5

Circuit components - section7.2.2, HW7Q12, lecture20

wires and buses (bus = collection of wires) - section7.2.2, HW7Q7, HW7Q8 memories - section7.2.2

arithmetic logic unit (ALU) - section7.2.2, HW7Q8

multiplexer (mux) - section7.2.2, HW7Q7, HW7Q8, lecture22slide12-19

program memory (PM) - section7.4.1

random access memory (RAM) - section7.4.1

register file (RF) - section7.4.1

program counter (PC), status register (SREG), and auxiliary registers - section7.4.1, HW7Q4, HW7Q6, lecture19slide16, lecture23slide5, lectures25-27

color legend: book, homework, lecture

### Exam 3 Topics - Chapter 7 (continued)

Instruction Implementation

Idi - section7.3.1and7.4.2, HW7Q9, lecture19slide17-19, lecture20slide13-20 Id - section7.3.2and7.4.3, HW7Q9, lecture19slide20-22, lecture23slide6-20 st, add, sub, cp, subi, cpi, rjmp, breq - section7.3.3through7.4.4 even more - lecture26slide10-19

From lecture26slide19: implementing instructions will be on the exam exam won't ask you to implement any instruction that involves I/O registers State Transition - section7.2.1, HW7Q11, lecture22slide21-23, lecture24slide5-21, lectures25-27 Tracing Instruction Execution in Hardware - chapter7, HW7Q11, HW7Q12, lectures25-27

#### **MUX: Notation & Selection**

Use the selector to indicate which input should be used as output.



### **AUX registers**



- Auxiliary registers have three ports: din, dout, and we
- When **we** is low (0) **dout** ignores **din**, and continues to be set to what **din** was before **we** went low.
- When we is high (1) dout = din

WE	0	1	0
DIN	1	1	0
DOUT			

#### Binary -> ISA

# 111000000001111 -> ldi ?, ? What do I need to look for?

### Binary -> ISA

- 1. Find the instruction format: CCCC\_IIII\_RRRR\_IIII
- **2**. **Divide binary**: 1110\_0000\_0000\_1111
- 3. Find the opcode: 1110
- 4. Calculate:
  - a. 0000 1111 -> 15
  - b. 0000 -> 0 -> r16
- 5. Instruction: r16, 15



# add r17, r19 How do we convert this to binary?

### ISA -> Binary

- 1. Look up the format: CCCCCC\_S\_RRRRR\_SSSS
- 2. Look up the opcode: 000011
- 3. Convert C, R, S
  - a. C = 00011
  - b. R = r17 = 10001
  - c. S = r19 = 10011
- 4. Insert: 000011\_1\_10001\_0011 -> 0000\_1111\_0001\_0011

# Tracing

; assume this is the entire program,

- ; so r13, r26 and r27 is initially 0
- ld r27, X ; demonstrates accessing RAM
- cp r13, r27 ; demonstrates updating SREG
- rjmp -23 ; demonstrates negative immediates and SEXT

; this is a bad program, so



# Tracing Id Encoding

Tracing in Lincounty	1d st	load from RAM	5-bit reg	1001000_RRRRR_1100	ld r1,	X = 1001000_r1_1100	= 1001000_00001
	Comment of the						
	Computatio	add without canny	E hit nog E hit nog	GOGGIII C DDDDD CCCC	add a1		- 000011 0 00001 0010
	sub	subtract without carry	S-bit reg. S-bit reg	000011_3_NNNNN_33333	sub n1	$n^2 = 000011[n^2[4][n^1[n^2[3:0]]]$	- 000011_0_00001_0010
	inc	increment	5-hit reg	1001010 RRRR 0011	inc r1	= 1001010 c1 0011	= 1001010 00001 0011
- ,	dec	decrement	5-bit reg	1001010 RRRRR 1010	dec r1	= 1001010 r1 1010	= 1001010 00001 1010
	and	logical AND	5-bit reg, 5-bit reg	001000 S RRRRR SSSS	and r1,	r2 = 001000 r2[4] r1 r2[3:0]	= 001000 0 00001 0010
27 - 16 + 9 + 7 + 1	or	logical OR	5-bit reg, 5-bit reg	001010 S_RRRRR_SSSS	or r1,	r2 = 001010_r2[4]_r1_r2[3:0]	= 001010_0_00001_0010
	eor	logical exclusive-OR	5-bit reg, 5-bit reg	001001_S_RRRRR_SSSS	eor r1,	r2 = 001001_r2[4]_r1_r2[3:0]	= 001001_0_00001_0010
	com	logical complement; NOT	5-bit reg	1001010_RRRRR_0000	com r1	= 1001010_r1_0000	= 1001010_00001_0000
	neg	negate	5-bit reg	1001010_RRRRR_0001	neg r1	= 1001010_r1_0001	= 1001010_00001_0001
- 0h11011	asr	arithmetic shift right	5-bit reg	1001010_RRRRR_0101	asr r1	= 1001010_r1_0101	= 1001010_00001_0101
	ср	compare	5-bit reg, 5-bit reg	000101_S_RRRRR_SSSS	cp r1,	r2 = 000101_r2[4]_r1_r2[3:0]	= 000101_0_00001_0010
$\sim$	subi	subtract immediate	4-bit reg, 8-bit imm	0101_IIII_RRRR_IIII	subi r16,	45 = <b>0101_</b> 45[7:4]_r <b>16_</b> 45[3:0]	= <b>0101</b> _0010_0000_1101
	andi	logical AND immediate	4-bit reg, 8-bit imm	0111_IIII_RRRR_IIII	andi r16,	45 = <b>0111_</b> 45[7:4]_r <b>16_</b> 45[3:0]	= <b>0111_</b> 0010_0000_1101
	ori	logical OR immediate	4-bit reg, 8-bit imm	0110_IIII_RRRR_IIII	ori r16,	45 = 0110_45[7:4]_r16_45[3:0]	= 0110_0010_0000_1101
	срі	compare with immediate	4-bit reg, 8-bit imm	0011_1111_RRRR_1111	cp1 r1/,	$45 = 0011_45[7:4]_{17}[45[3:0]$	= 0011_0010_0001_1101
	auc	add with carry	S-bit reg, S-bit reg	000111_5_RRKRK_5555	abc r1,	PZ = 000111 PZ[4] P1 PZ[3:0]	= 000111_0_00001_0010
	SUC	subchace with carry	5-bit reg, 5-bit reg	000010_2_VVVVV_2222	SUC P1,	12 = 000010_12[4]_11_12[5:0]	= 000010_0_00001_0010
	Input/Outp	ut Instructions		COMPANY STATES AND STATES			
	in	load from I/O register	5-bit reg, I/O reg	10110_AA_RRRRR_AAAA	in r1, 16	= 10110_I016[5:4]_r1_I016[3:0]	= 10110_01_00001_0000
	out	store to I/O register	5-bit reg, I/O reg	10111_AA_RRRRR_AAAA	out 18, r1	= 10111_I018[5:4]_r1_I018[3:0]	= 10111_01_00001_0010
	Control_F	ow Instructions					
	rimp	relative jump	12-bit imm	1100 TTTTTTTTTTTT	rimp 4	= 1100 4	= 1100 000000000100
	breg	branch if equal	7-bit imm	111100 IIIIII 001	breg 2	= 111100 2 001	= 111100 0000010 001
	brne	branch if not equal	7-bit imm	111101_IIIIIII_001	brne 2	= 111101_2_001	= 111101_0000010_001
	brsh	branch if same/higher	7-bit imm	111101_IIIIIII_000	brsh 2	= 111101_2_000	= 111101_0000010_000
	brlo	branch if lower	7-bit imm	111100_IIIIII_000	brlo 2	= 111100_2_000	= 111100_0000010_000
	rcall	relative call subroutine	12-bit imm	1101_IIIIIIIIII	rcall -23	= <b>1101</b> 23	= <b>1101</b> _111111101001
	ret	return from subroutine	other	1001_0101_0000_1000	ret	= 1001_0101_0000_1000	= 1001_0101_0000_1000
	Stack Inst	ructions	1				
	push	push register to stack	5-bit reg	1001001_RRRRR_1111	push r1	= 1001001_r1_1111	= 1001001_00001_1111
	рор	pop register off stack	5-bit reg	1001000 RRRRR_1111	pop r1	= 1001000_r1_1111	= 1001000_00001_1111
			-	1-	-		
	Format Leg	gend:	Туре	Format	4		
	underscore	s are used for readability	5-bit reg	CCCCCCC RRRRR CCCC			

Instruction Description

load immediate

Load/Store Instructions

ldi

1d

Type

4-bit reg, 8-bit imm

5-bit reg, 5-bit reg

4-bit reg, 8-bit imm

5-bit reg, I/O reg

7-bit imm

12-bit imm

Encoding

1110\_IIII\_RRRR\_IIII

CCCCCC S RRRRR SSS

CCCCC AA RRRRR AAAA

CCCC IIII RRRR III

CCCCC TITITI CCC

CCCC IIIIIIIIIIIIII

Example

ldi r16, 45 = 1110\_45[7:4]\_r16\_45[3:0] = 1110\_0010\_0000\_1101

other 2222\_2222\_2222\_2222 Encoding Instruction Description Type Example ld 5-bit reg load from RAM 1001000\_RRRRR\_1100 1d r1, X = 1001000 r1 1100= 1001000 00001 1100

= opcode

first register

diate

hd register

register

 $INST = 0b100100_{11011} 1100 = 0b1001 0001 1011 1100 = 0x91BC = 37308^{16}$ 

### Tracing Id Cycle 1

INST =  $0b100100_{11011_{1100}} = 0b1001_{0001_{1011_{1100}}} = 0x91BC = 37308$ 

Cycle	State	Changed registers/values and control signals	INST=PM[PC]			din dout	addr dout	din dout
1	00000	<pre>INST = 0x91BC, INST_we = 1, other_ctrls = 0</pre>	breg Idi,sub 16(INST[9:3]):0 00011	REG=1,INST[7:4]	REG=INST[8:4] 00010	PC		
2						+1	РМ	
3								
4								
5						DC 140	INST	[ <u>we</u> REG_se]
6								17



# Tracing Id Cycle 3

Cycle

State



# Tracing Id Cycle 4

; assume this is the entire program,

so no assembler directive and all RAM is 0

![](_page_19_Figure_3.jpeg)

# Tracing Id Cycle 5 VAL = 0, REG = 27

Cycle	State	Changed registers/values and control signals
1	00000	<pre>INST = 0x91BC, INST_we = 1, other_ctrls = 0</pre>
2	00010	<pre>REG = 27, REG_we = 1, REG_sel = 1, other_ctrls = 0</pre>
3	01001	ADDR = 0, ADDR_we = 1, other_ctrls = 0
4	10000	<pre>VAL = 0, VAL_we = 1, VAL_sel = 0, other_ctrls = 0</pre>
5	10010	r27 = 0, RF_we = 1, RF_sel = 0, other_ctrls = 0
6		

![](_page_20_Figure_2.jpeg)

![](_page_20_Figure_3.jpeg)

![](_page_20_Figure_4.jpeg)

Т	racii	ng ld Cycle 6	PC = 0 PC = PC + 1 = 0 + 1 = 1	din dout a
Cycle	State	Changed registers/values and control signals		РС
1	00000	<pre>INST = 0x91BC, INST_we = 1, other_ctrls = 0</pre>	RF[REG]=VAL	
2	00010	<pre>REG = 27, REG_we = 1, REG_sel = 1, other_ctrls = 0</pre>	Idi,Id,add,sub,subi	
3	01001	ADDR = 0, ADDR_we = 1, other_ctrls = 0	PC=PC+1	
4	10000	<pre>VAL = 0, VAL_we = 1, VAL_sel = 0, other_ctrls = 0</pre>	10011	
5	10010	r27 = 0, RF_we = 1, RF_sel = 0, other_ctrls = 0		PC_sel
6	10011	<pre>PC = 1, PC_we = 1 PC_sel = 1, other_ctrls = 0</pre>		PC_we inst_in

	Instructio	on Description	Туре	Encoding	Example	
	Load/Store	load immediate	4-hit reg. 8-hit imm	1110 TITT RRRR TITT	ldi r16, 45 = 1110 45[7:4] r16 45[3:0]	= 1110 0010 0000 1101
I racing on Encoding	mov	move; copy register	5-bit reg, 5-bit reg	001011_S_RRRRR_SSSS	mov r1, r2 = 001011_r2[4]_r1_r2[3:0]	= 001011_0_00001_0010
	ld	load from RAM	5-bit reg	1001000_RRRRR_1100	ld r1, X = 1001000_r1_1100	= 1001000_00001_1100
	st	store to RAM	5-bit reg	1001001_RRRRR_1100	st X, r1 = 1001001_r1_1100	= 1001001_00001_1100
	Computatio	on Instructions				
1007	add	add without carry	5-bit reg, 5-bit reg	000011_S_RRRRR_SSSS	add r1, r2 = 000011_r2[4]_r1_r2[3:0]	= <b>000011_0_00001_0010</b>
cp rl3, r2/	sub	subtract without carry	5-bit reg, 5-bit reg	000110_S_RRRRR_SSSS 1001010_RRRRR_0011	sub r1, r2 = 000110_r2[4]_r1_r2[3:0]	= 000110_0_00001_0010 = 1001010_00001_0011
•F ==•, ==·	dec	decrement	5-bit reg	1001010_RRRRR_1010	dec r1 = 1001010_r1_1010	= 1001010_00001_1010
	and	logical AND	5-bit reg, 5-bit reg	001000_S_RRRRR_SSSS	and r1, r2 = 001000_r2[4]_r1_r2[3:0]	= 001000_0_00001_0010
13 = 8 + 4 + 1	epr	logical OR logical exclusive-OR	5-bit reg, 5-bit reg	001010_S_RRRRR_SSSS 001001_S_RRRRR_SSSS	or r1, r2 = $001010 r2[4] r1 r2[3:0]$ eor r1, r2 = $001001 r2[4] r1 r2[3:0]$	= 001010_0_00001_0010 = 001001_0_00001_0010
	com	logical complement; NOT	5-bit reg	1001010_RRRRR_0000	com r1 = 1001010_r1_0000	= 1001010_00001_0000
01 01 1 01	neg	negate	5-bit reg	1001010_RRRRR_0001	neg r1 = 1001010_r1_0001	= 1001010_00001_0001
= 10110d0 =	ср	compare	5-bit reg, 5-bit reg	000101_S_RRRRR_SSSS	<pre>cp r1, r2 = 000101_r2[4]_r1_r2[3:0]</pre>	= 000101_0_00001
	andi	logical AND immediate	4-bit reg, 8-bit imm	0111_IIII_RRRR_IIII	andi r16, 45 = 0111_45[7:4]_r16_45[3:0]	= 0111_0010_0000_1101
27 = 16 + 8 + 2 + 1	ori	logical OR immediate	4-bit reg, 8-bit imm	0110_IIII_RRRR_IIII	ori r16, 45 = 0110_45[7:4]_r16_45[3:0]	= 0110_0010_0000_1101
	cpi adc	compare with immediate add with carry	4-bit reg, 8-bit imm 5-hit reg, 5-hit reg	0011_1111_RRRR_1111	cpi r17, 45 = $0011_{45}[7:4]_{r17}[45[3:0]]$ adc r1, r2 = $000111_{r2}[4]_{r1}[r2[3:0]]$	= 0011_0010_0001_1101 = 0001111_0_00001_0010
	sbc	subtract with carry	5-bit reg, 5-bit reg	000010_S_RRRRR_SSSS	sbc r1, r2 = 000010_r2[4]_r1_r2[3:0]	= 000010_0_00001_0010
= 0 h 1 1 0 1 1	Travet (Out	Tastaustiene				N
	in	load from I/O register	5-bit reg, I/O reg	10110_AA_RRRRR_AAAA	in r1, 16 = 10110_I016[5:4]_r1_I016[3:0	e] = 10110_01_00001_0000
	out	store to I/O register	5-bit reg, I/O reg	10111_AA_RRRRR_AAAA	out 18, r1 = 10111_I018[5:4]_r1_I018[3:6	ð] = 10111_01_00001_0010
	Control-F	low Instructions				
	rjmp	relative jump	12-bit imm	1100_IIIIIIIIIII	rjmp 4 = 1100_4	= <b>1100</b> _00000000100
	breq	branch if equal	7-bit imm 7-bit imm	111100_IIIIIII_001	breq 2 = 111100_2_001	= 111100_0000010_001
	brsh	branch if same/higher	7-bit imm	111101_IIIIII_000	brsh 2 = 111101 2 000	= 111101_0000010_000
	brlo	branch if lower	7-bit imm	111100_IIIIIII_000	brlo 2 = 111100_2_000	= 111100_0000010_000
	rcall	relative call subroutine	12-bit imm		rcall -23 = 110123	= 1101_11111101001 - 1001_0101_0000_1000
		Sub-Sub-Sub-Sub-Sub-Sub-Sub-Sub-Sub-Sub-	o cher	1001_0101_0000_1000		- 1001_0101_0000_1000
	Stack Inst	tructions	It has an	4004004 00000 4444		4004004 00004 4444
	pusn	pop register off stack	5-bit reg	1001001_KRKRR_1111 1001000 RRRRR 1111	pop r1 = 1001000 r1 1111	= 1001001_00001_1111
		-	-	1-	7	
	escore	gend: es are used for readability	Type 5-hit reg	Format	-	
	pde	e	5-bit reg, 5-bit reg	CCCCCC_S_RRRRR_SSSS	3	,
		register	5-bit reg, I/O reg	CCCCC_AA_RRRRR_AAAA	-	
			7-bit imm		-	//
			12-bit imm	CCCC_IIIIIIIIIIII		//
			other	uu_uu_uu	<b>_</b>	
Instruction Description Type	Encourse	Examp	10			
	Encoulin		10 000			
cp  compare  5-bit reg, 5-bit re	g 000101_5	_RRRRR_SSSS CP	$r_1, r_2 = 000$	101_r2[4]_r1	_r2[3:0] = 000101_0_	00001_0010
٨						
TNOM $- 0 - 0 - 0 - 1 - 1 - 1 - 1 - 1 - 1 - 1 $	∩1_	0001 01	10 110	1 1 0 1	1 _ 01(DD	_ EOE1 <sup>23</sup>
1101 10110 1 1010000 = 12011	= UD	JUUUT UT	TO TTC	L T O T	T = 0 X T 0 D B	= 3031
		—	—			

#### Tracing cp Cycle 1 INST = 0b0001\_0110\_1101\_1011

Cycle	State	Changed registers/values and control signals	$= 0 \times 16 DB$
1	00000	<pre>INST = 0x16DB, INST_we = 1, other_ctrls = 0</pre>	INST=PM[PC] 00000 breq Ildi,subi,cpi Id,st,add,sub,cp PC INST
2			16(INST[9:3]):0 REG=1,INST[7:4] 00001 +1 PM
3			
4			
5			
6			INST_we REG C_we R

![](_page_24_Figure_1.jpeg)

INST[15], INST[8] INST[4] INST[0]

assume this is the entire program, so r13, r26 and r27 is initially 0

![](_page_25_Figure_2.jpeg)

;

INST[15], INST[9] INST[3] INST[0]

Cycle State Changed registers/values and INST = INST[15:0] = 0b0001 0110 1101 1011control signals INST[9], INST[8:4] = 0b1, 0b1011 = 0b1101100000  $INST = 0 \times 16 DB$ , REG2 = INST[9], INST[8:4] = 0b11011 = 27INST we = 1, other ctrls = 0din dout 2 00010 REG = 13, REG we = 1, REG REG2=INST[9],INST[3:0] we REG sel = 1, INST other ctrls = 0din dout 3 00111 VAL1 = 0, VAL1 we = 1, add,sub,cp RF sel = 0, REG2 other ctrls = 001010 REG2 = 27, REG2 we = 1, 4 VAL2=RF[REG2] other ctrls = 0OFF 5 6 we REG sel RF OFF sel REG we OFF we 27 REG2 we

Cycle	State	Changed registers/values and control signals
1	00000	<pre>INST = 0x16DB, INST_we = 1, other_ctrls = 0</pre>
2	00010	<pre>REG = 13, REG_we = 1, REG_sel = 1, other_ctrls = 0</pre>
3	00111	<pre>VAL1 = 0, VAL1_we = 1, RF_sel = 0, other_ctrls = 0</pre>
4	01010	REG2 = 27, REG2_we = 1, other_ctrls = $0$
5	01000	<pre>VAL2 = 0, VAL2_we = 1, VAL2_sel = 0, RF_sel = 1, other_ctrls = 0</pre>
6		

assume this is the entire program, so r13, r26 and r27 is initially 0

REG2 = 27 VAL2 = RF[REG2] = RF[27] = 0  $\boxed{[11:8],INST[3:0]}$  VAL2=RF[REG2] 01000

VAL=VAL1+VAL2

01011

VAL=VAL1-VAL2

01100

;

![](_page_27_Figure_4.jpeg)

Т	raci	ng cp Cycle 6	
Cycle	State	Changed registers/values and control signals	$\begin{bmatrix} VALI &= 0, & VAL2 &= 0 \\ VAL &= VAL1 - VAL2 \end{bmatrix}$
1	00000	<pre>INST = 0x16DB, INST_we = 1, other_ctrls = 0</pre>	= 0 - 0 = 0
2	00010	<pre>REG = 13, REG_we = 1, REG_sel = 1, other_ctrls = 0</pre>	VAL=VAL1-VAL2     VAL=VA     VAL1     B     Sreg_out     RAM       subi,cpi,sub,cp     add     we     op     op
3	00111	<pre>VAL1 = 0, VAL1_we = 1, RF_sel = 0, other_ctrls = 0</pre>	Update SREG
4	01010	<pre>REG2 = 27, REG2_we = 1, other_ctrls = 0</pre>	01101
5	01000	<pre>VAL2 = 0, VAL2_we = 1, VAL2_sel = 0, RF_sel = 1, other_ctrls = 0</pre>	A_sel RAM_we VAL_sel B sel VAL_sel VAL_sel
6	01100	<pre>VAL = 0, VAL_we = 1, VAL_sel = 1, A_sel = 1, B_sel = 0, ALU_op = 1 other_ctrls = 0</pre>	we ADDR_we SREG_we VAL_we

AL

29

Т	racii	ng cp Cycle 7		A Q-	
Cycle	State	Changed registers/values and control signals	ALU: $op2 - op1 = result$ op2 = 0, op1 = 0, result = 0	- B <sup>Sreg_out</sup> - sregin	RAM
			N set if negative	ор	•
4	01010	<pre>REG2 = 27, REG2_we = 1, other_ctrls = 0</pre>	N = 0  (result == 0) $Z  set if result == 0$	ALU	
5	01000	<pre>VAL2 = 0, VAL2_we = 1, VAL2_sel = 0, RF_sel = 1, other_ctrls = 0</pre>	Z = 1  (result == 0) C set if op1 < op2 (2's comp)		we SREG
6	01100	<pre>VAL = 0, VAL_we = 1, VAL_sel = 1, A_sel = 1, B_sel = 0, ALU_op = 1 other_ctrls = 0</pre>	C = 0 (op1 == op2 == 0)	Ł	[11:8],
7	01101	<pre>update SREG (Z=1,C=0, N=0), SREG_we = 1, other_ctrls = 0</pre>	add,sub,subi		
8			p, breq	RAM_we	we
			PC=PC+1		30

Cycle	State	Changed registers/values and control signals	PC = 1
•••			PC = PC + 1 = 1 + 1 = 2
4	01010	<pre>REG2 = 27, REG2_we = 1, other_ctrls = 0</pre>	
5	01000	<pre>VAL2 = 0, VAL2_we = 1, VAL2_sel = 0, RF_sel = 1, other_ctrls = 0</pre>	RF[REG]=VAL 10010
6	01100	<pre>VAL = 0, VAL_we = 1, VAL_sel = 1, A_sel = 1, B_sel = 0, ALU_op = 1 other_ctrls = 0</pre>	Idi,Id,add,sub,subi
7	01101	update SREG (Z=1,C=0, N=0), SREG_we = 1, other_ctrls = 0	PC=PC+1 10011 ←
8	10011	<pre>PC = 2, PC_we = 1, PC_sel = 0, other_ctrls = 0</pre>	

![](_page_30_Figure_2.jpeg)

		Instruction	Description	Туре	Encoding	Example			
		Load/Store In	Istructions	4 hit ong 9 hit imm	1110 TITT DDDD TITT	Idi a16	45 - 1110 4517-41 -16 4512-01	- 1110 0010 0000 1101	
I racina rimn H	ncodina	mov	move; copy register	5-bit reg, 5-bit reg	001011_S_RRRRR_SSSS	mov r1,	r2 = 001011_r2[4]_r1_r2[3:0]	= 001011_0_00001_0010	
	ILCUUIIIG	ld	load from RAM	5-bit reg	1001000_RRRRR_1100	ld r1,	X = 1001000_r1_1100	= 1001000_00001_1100	
	0	st	store to RAM	5-bit reg	1001001_RRRRR_1100	st X,	r1 = 1001001_r1_1100	= 1001001_00001_1100	
		Computation :	Instructions						
		add	add without carry	5-bit reg, 5-bit reg	000011_S_RRRRR_SSSS	add r1,	r2 = 000011_r2[4]_r1_r2[3:0]	= 000011_0_00001_0010	
r-23		sub	subtract without carry	5-bit reg, 5-bit reg	000110_S_RRRRR_SSSS	sub r1,	r2 = 000110_r2[4]_r1_r2[3:0]	= 000110_0_00001_0010	
		dec	decrement	5-bit reg	1001010_RRRRR_1010	dec r1	= 1001010_r1_1010	= 1001010_00001_1010	
		and	logical AND	5-bit reg, 5-bit reg	001000_S_RRRRR_SSSS	and r1,	r2 = 001000_r2[4]_r1_r2[3:0]	= 001000_0_00001_0010	
23 = 16 + 4 + 2	+ 1	or	logical OR	5-bit reg, 5-bit reg	001010_S_RRRRR_SSSS	or r1,	r2 = 001010 r2[4] r1 r2[3:0]	= 001010_0_00001_0010 - 001001_0_00001_0010	
		com	logical complement; NOT	5-bit reg	1001010_RRRRR_0000	com r1	= 1001010_r1_0000	= 1001010_00001_0000	
	0111	neg	negate	5-bit reg	1001010_RRRRR_0001	neg r1	= 1001010_r1_0001	= 1001010_00001_0001	
= (b)(0)(0)(0)(0)(0)	()111	asr	arithmetic shift right	5-bit reg	1001010_RRRRR_0101	asr r1	$= 1001010 r1_0101$	= 1001010_00001_0101	
	- <u> </u>	subi	subtract immediate	4-bit reg, 8-bit imm	0101 IIII RRRR IIII	subi r16.	45 = 0101 45[7:4] r16 45[3:0]	= 0101 0010 0000 1101	
the second letter and a shall of the	a.e.t. 00	andi	logical AND immediate	4-bit reg, 8-bit imm	0111_IIII_RRRR_IIII	andi r16,	45 = 0111_45[7:4]_r16_45[3:0]	= 0111_0010_0000_1101	
invert bits and add 1 to	get -23	ori	logical OR immediate	4-bit reg, 8-bit imm	0110_IIII_RRRR_IIII	ori r16,	45 = 0110_45[7:4]_r16_45[3:0]	= 0110_0010_0000_1101	
	0	adc	compare with immediate	4-bit reg, 8-bit imm	0011_1111_RRRR_1111 0001111_S_RRRRR_SSSS	cpi r1/,	$45 = 0011_45[7:4]_r17_45[3:0]$ $r2 = 000111_r2[4]_r1_r2[3:0]$	= 0011_0010_0001_1101 = 000111_0_00001_0010	
00 01 1111 1110	100011	sbc	subtract with carry	5-bit reg, 5-bit reg	000010_S_RRRRR_SSSS	sbc r1,	r2 = 000010_r2[4]_r1_r2[3:0]	= 000010_0_00001_0010	
-23 = 001111 1110	J IUUU+I			0.0004 20.00					
—	—	in Input/Output	load from I/O register	5-hit reg. T/O reg	10110 AA RRRRR AAAA	in c1, 16	= 10110 T016[5:4] c1 T016[3:0]	= 10110 01 00001 0000	
01-1111 1110	1001	out	store to I/O register	5-bit reg, I/O reg	10111_AA_RRRRR_AAAA	out 18, r1	= 10111_I018[5:4]_r1_I018[3:0]	= 10111_01_00001_0010	
)	J TOOT			10.000 Mills 1000	10.5 5.61 0.53				
<u> </u>	— <u> </u>	rimp	relative jump	12-hit imm	1100 TITTTTTTTTT	rimp 4	= 1100 4	= 1100 0000000010	
		breg	pranch II cquai	7-DIC INN	111100_111111_001	Died 2	= 11100_2_001	- 111100_0000010_001	
		brne	branch if not equal	7-bit imm	111101_IIIIII_001	brne 2	= 111101_2_001	= 111101_0000010_001	
		brio	branch if lower	7-bit imm	111101_1111111_000 111100_11111111_000	brlo 2	= 111101_2_000	= 111101_0000010_000	- )
		rcall	relative call subroutine	12-bit imm	<b>1101_</b> IIIIIIIIIII	rcall -23	<b>= 1101_</b> -23	= <b>1101</b> _111111101001	
		ret	return from subroutine	other	1001_0101_0000_1000	ret	= 1001_0101_0000_1000	= 1001_0101_0000_1000	
		Stack Instru	tions						
		push	push register to stack	5-bit reg	1001001_RRRRR_1111	push r1	= 1001001_r1_1111	= 1001001_00001_1111	
		pop	pop register off stack	5-bit reg	1001000_RRRRR_1111	pop r1	= 1001000_r1_1111	= 1001000_00001_1111	
		Format Legend	1:	Туре	Format	٦			
		underscores_a	are_used_for_readability	5-bit reg	CCCCCCC_RRRRR_CCCC				
		= opcode	riston	5-bit reg, 5-bit reg	CCCCCC_S_RRRRR_SSSS	-			
		d re	egister	4-bit reg, 8-bit imm	CCCC_IIII_RRRR_IIII	1			
		egi	ster	7-bit imm	CCCCCC_IIIIIII_CCC				
		liate		12-bit imm		-			
	/			orner	the the the the	4			
Instruction Description	vne En	coding	Examp	10					//
	ype En	couring	Lvamp	Te				——————————————————————————————————————	
rjmp relative jump 12	2-bit imm 11	<u>00_11111</u>	IIIIIII rjmp	4 = 110	0_4		= 1100_00000	0000100	
									0
TNC	T - 0 h 1 1 0	$\cap$ 1 <sup>-1</sup>	111 111	0 1 0 0	1 - 0	VCL		53005	3
TNS	$ 0$ $\pm$ $1$	U I.		LO TOO	$\perp - 0$	XUI	<u> に ッ ― し</u> 00		
		—		_					

#### Tracing rjmp Cycle 1 INST = 0b1100 1111 1110 1001 = 0xCFE9 = 0d53225

![](_page_32_Figure_1.jpeg)

![](_page_33_Figure_0.jpeg)

# Tracing rjmp Cycle 3

Gelse if subtraction, then ALU\_op = 0 else if subtraction, then ALU\_op = 1 OFF = -23; PC = 2

![](_page_34_Figure_2.jpeg)

Т	racir	ng rjmp Cycle	4			
			VAL = -21		din dout 2	addr
Cycle	State	Changed registers/values and control signals	PC = VAL = -21		PC	
1	00000	<pre>INST = 0xCFE9, INST_we = 1, other_ctrls = 0</pre>	PC=VAL 10100 rjmp, breq Idi,Id,add,sub,s		-+1	
2	00100	OFF = $0 \times FFE9 = -23$ , OFF_we = 1, OFF_sel = 0, other_ctrls = 0	PC=PC+1 10011		, l	PN
3	01110	<pre>VAL = -21, VAL_we = 1, VAL_sel = 1, A_sel = 0, B_sel = 1, ALU_op = 0, other_ctrls = 0</pre>				
4	10100	<pre>PC = -21, PC_we = 1 PC_sel = 0, other_ctrls = 0</pre>		Pr Co	1	
5				in	PC_we	

# Tracing rjmp Cycle 5

	PC = -21	din dout addr	
values and	PC = PC + 1 = -21 + 1 = -20	PC	
1	PC=VAL 10100 RF[REG]=VAL 10010		
0	rjmp, bred		
-23,	PC=PC+1 10011		
0			
_we = 1, _sel = 0, _op = 0,			
e = 1		¢	
)		PC_sel	
e = 1		PC_we	
)		Linst_in 3	37

Cycle	State	Changed registers/values and control signals
1	00000	<pre>INST = 0xCFE9, INST_we = 1, other_ctrls = 0</pre>
2	00100	OFF = $0xFFE9 = -23$ , OFF_we = 1, OFF_sel = 0, other_ctrls = 0
3	01110	<pre>VAL = -21, VAL_we = 1, VAL_sel = 1, A_sel = 0, B_sel = 1, ALU_op = 0, other_ctrls = 0</pre>
4	10100	<pre>PC = -21, PC_we = 1 PC_sel = 0, other_ctrls = 0</pre>
5	10011	<pre>PC = -20, PC_we = 1 PC_sel = 1, other_ctrls = 0</pre>