Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clks
NOP		No Operation			1
ADD	Rd, Rs1, Rs2	Add two Registers	Rd ← Rs1 + Rs2	C,O,N,Z	1
ADDI	Rd, Rs1, #data	Add Register with Immediate	Rd ← Rs1 + #data	C,O,N,Z	1
ADC	Rd, Rs1, Rs2	Add two Registers with Carry	Rd ← Rs1 + Rs2 + C	C,O,N,Z	1
ADCI	Rd, Rs1, #data	Add Register with Immediate with Carry	Rd ← Rs1 + #data + C	C,O,N,Z	1
SUB	Rd, Rs1, Rs2	Subtract two Registers	Rd ← Rs1 - Rs2	C,O,N,Z	1
SUBI	Rd, Rs1, #data	Subtract Immediate from Register	Rd ← Rs1 - #data	C,O,N,Z	1
SBB	Rd, Rs1, Rs2	Subtract two Registers with Borrow	Rd ← Rs1 - Rs2 - C	C,O,N,Z	1
SBBI	Rd, Rs1, #data	Subtract Immediate from Register with Borrow	Rd ← Rs1 - #data - C	C,O,N,Z	1
CMP	Rs1, Rs2	Compare two Registers	Rs1 - Rs2	C,O,N,Z	1
CMPI	Rs1, #data	Compare Register with Immediate	Rs1 - #data	C,O,N,Z	1
AND	Rd, Rs1, Rs2	Logical AND Registers	Rd ← Rs1 • Rs2	N,Z	1
ANDI	Rd, Rs1, #data	Logical AND Register with Immediate	Rd ← Rs1 • #data	N,Z	1
OR	Rd, Rs1, Rs2	Logical OR Registers	Rd ← Rs1 v Rs2	N,Z	1
ORI	Rd, Rs1, #data	Logical OR Register with Immediate	Rd ← Rs1 v #data	N,Z	1
XOR	Rd, Rs1, Rs2	Logical XOR Registers	Rd ← Rs1 ⊕ Rs2	N,Z	1
XORI	Rd, Rs1, #data	Logical XOR Register with Immediate	Rd ← Rs1 ⊕ #data	N,Z	1
SHR	Rd, Rs1, Rs2	Logical or Arithmetic Shift Right	Rd ← Rs1 >> Rs2[3:0]	C,N,Z	1
SHRI	Rd, Rs1, #am	Logical or Arithmetic Shift Right Immediate Amount	Rd ← Rs1 >> #am	C,N,Z	1
SHL	Rd, Rs1, Rs2	Logical or Arithmetic Shift Left	Rd ← Rs1 << Rs2[3:0]	C,O,N,Z	1
SHLI	Rd, Rs1, #am	Logical or Arithmetic Shift Left Immediate Amount	Rd ← Rs1 << #am	C,O,N,Z	1
COM	Rd, Rs1	One's Complement	Rd ← #FFFF – Rs1	N,Z	1
NEG	Rd, Rs1	Two's Complement	Rd ← #0000 – Rs1	O,N,Z	1
ROL	Rd, Rs1	Rotate Left through Carry	Rd[0]←C,Rd[n+1]←Rs1[n],C←Rs1[15]	C,N,Z	1
ROR	Rd, Rs1	Rotate Right through Carry	Rd[15]←C,Rd[n]←Rs1[n+1],C←Rs1[0]	C,N,Z	1
MOV	Rd, Rs1	Load from Register to Register	Rd ← Rs1		1
MOVI	Rd, #data	Load Immediate to Register	Rd ← #data		1
MOVF	Rd, Rs1	Load from Register to Register with Test	Rd ← Rs1	N,Z	1
MULT	Rd, Rs1, Rs2	Multiply two Registers	Rd ← Rs1 x Rs2	C,O,N,Z	10
MULTI	Rd, Rs1, #data	Multiply Register with Immediate	Rd ← Rs1 x #data	C,O,N,Z	10
DIV	Rd, Rs1, Rs2	Divide two Registers	Rd ← Rs1 / Rs2, Rd+1← Rs1 % Rs2	C,O,N,Z	19
DIVI	Rd, Rs1, #data	Divide Register by Immediate	Rd ← Rs1 / #data, Rd+1← Rs1 % #data	C,O,N,Z	19
SET	#mask	Set or Clear Flags by Mask	Flags← {Flags • #maskMsb} v #maskLsb		
	#mask = 0xFF80	Set Masking Flag	M ← 1	M	1
	#mask = 0x7F00	Clear Masking Flag	M ← 0	M	1
	#mask = 0xFF40	Set Enable Flag	E ← 1	E	1
	#mask = 0xBF00	Clear Enable Flag	E ← 0	E	1
	#mask = 0xFF20	Set Interrupt Flag	I ← 1	I	1
	#mask = 0xDF00	Clear Interrupt Flag	I ← 0	I	1
	#mask = 0xFF10	Set Signed Flag	S ← 1	S	1
	#mask = 0xEF00	Clear Signed Flag	S ← 0	S	1
	#mask = 0xFF08	Set Carry Flag	C ← 1	C	1
	#mask = 0xF700	Clear Carry Flag	C ← 0	C	1
	#mask = 0xFF04	Set Overflow Flag	O ← 1	O	1
	#mask = 0xFB00	Clear Overflow Flag	O ← 0	O	1
	#mask = 0xFF02	Set Negative Flag	N ← 1	N	1
	#mask = 0xFD00	Clear Negative Flag	N ← 0	N	1
	#mask = 0xFF01	Set Zero Flag	Z ← 1	Z	1
	#mask = 0xFE00	Clear Zero Flag	Z ← 0	Z	1

Mnemonics	Operands	Description	Operation	Flags	#Clks
LOAD	Rd, Rs1	Load Indirect from RAM	Rd ← (Rs1)		1
LOADI	Rd, #addr	Load Direct from RAM	Rd ← (#addr)		1
STORE	Rs1, Rs2	Store Indirect to RAM	(Rs2) ← Rs1		1
STOREI	Rs1, #addr	Store Direct to RAM	(#addr) ← Rs1		1
IN	Rd, #sel	Load from Input Port	Rd ← Pin[#sel]		1
OUT	#sel, #op, Rs1	Store Register to Output Port with Operation	Port[#sel] ← #op(Rs1, Port[#sel])		
	#op = 0x0	Operation = NOP	Port[#sel] ← Rs1		1
	#op = 0x1	Operation = AND	Port[#sel] ← Rs1 • Port[#sel]		1
	#op = 0x2	Operation = OR	Port[#sel] ← Rs1 v Port[#sel]		1
	#op = 0x3	Operation = XOR	Port[#sel] ← Rs1 ⊕ Port[#sel]		1
OUTI	#sel, #op, #data	Store Immediate to Output Port with Operation	Port[#sel] ← #op(#data, Port[#sel])		
	#op = 0x0	Operation = NOP	Port[#sel] ← #data		1
	#op = 0x1	Operation = AND	Port[#sel] ← #data • Port[#sel]		1
	#op = 0x2	Operation = OR	Port[#sel] ← #data v Port[#sel]		1
	#op = 0x3	Operation = XOR	Port[#sel] ← #data ⊕ Port[#sel]		1
JMP	#addr	Direct Jump	PC ← #addr		2
JC	#addr	Jump If Carry	If (C == 1) then PC ← #addr		2
JNC	#addr	Jump If Not Carry	If (C == 0) then PC ← #addr		2
JO	#addr	Jump If Overflow	If (O == 1) then PC ← #addr		2
JNO	#addr	Jump If Not Overflow	If (O == 0) then PC ← #addr		2
JN	#addr	Jump If Negative	If (N == 1) then PC ← #addr		2
JP	#addr	Jump If Positive	If (N == 0) then PC ← #addr		2
JZ	#addr	Jump If Zero	If (Z == 1) then PC ← #addr		2
JNZ	#addr	Jump If Not Zero	If (Z == 0) then PC ← #addr		2
JL	#addr	Jump If Lower	If (C v (N⊕O) == 1) then PC ← #addr		2
JLE	#addr	Jump If Lower or Equal	If (Z v C v(N⊕O) == 1) then PC ← #addr		2
JG	#addr	Jump If Greater	If (Z v C v(N⊕O) == 0) then PC ← #addr		2
JGE	#addr	Jump If Greaterr or Equal	If (C v (N⊕O) == 0) then PC ← #addr		2
CALL	#addr	Subroutine Call	CALL_STACK ← PC + 1, PC ← #addr		2
RET		Subroutine Return	PC ← CALL_STACK		2
PUSH	Rs1	Push Register on Stack	DATA_STACK ← Rs1		1
POP	Rd	Pop Register from Stack	Rd ← DATA_STACK		1
PUSHF		Push Flags on Stack	DATA_STACK ← Flags		1
POPF		Pop Flags from Stack	Flags ← DATA_STACK		1
STOP		Stop Program Execution			1

Instruction Set Opcode Fields

NOP	
# 0x00	-
31 24	23 0

ADD				
# 0x0C	Rd	Rs1	Rs2	-
31 24	23 20	19 16	15 12	11 0

ADDI			
# 0x0E	Rd	Rs1	#data
31 24	23 20	19 16	15 0

ADC				
# 0x0D	Rd	Rs1	Rs2	-
31 24	23 20	19 16	15 12	11 0

ADCI			
# 0x0F	Rd	Rs1	#data
31 24	23 20	19 16	15 0

SUB				
# 0x1C	Rd	Rs1	Rs2	-
31 24	23 20	19 16	15 12	11 0

SUBI			
# 0x1E	Rd	Rs1	#data
31 24	23 20	19 16	15 0

SBB				
# 0x1D	Rd	Rs1	Rs2	-
31 24	23 20	19 16	15 12	11 0

SBBI							
# 0x1F		Rd		Rs1		#data	
31	24	23	20	19	16	15	0

CMP							
# 0x14		-		Rs1		Rs2	-
31	24	23	20	19	16	15 12	11 0

CMPI							
# 0x16		-		Rs1		#data	
31	24	23	20	19	16	15	0

AND							
# 0x2C		Rd		Rs1		Rs2	-
31	24	23	20	19	16	15 12	11 0

ANDI							
# 0x2E		Rd		Rs1		#data	
31	24	23	20	19	16	15	0

OR							
# 0x3C		Rd		Rs1		Rs2	-
31	24	23	20	19	16	15 12	11 0

ORI							
# 0x3E		Rd		Rs1		#data	
31	24	23	20	19	16	15	0

XOR							
# 0x4C		Rd		Rs1		Rs2	-
31	24	23	20	19	16	15 12	11 0

XORI							
# 0x4E		Rd		Rs1		#data	
31	24	23	20	19	16	15	0

SHR									
# 0x5C		Rd		Rs1		Rs2		-	
31	24	23	20	19	16	15	12	11	0

SHRI									
# 0x5E		Rd		Rs1		-		#am	
31	24	23	20	19	16	15	4	3	0

SHL									
# 0x6C		Rd		Rs1		Rs2		-	
31	24	23	20	19	16	15	12	11	0

SHLI									
# 0x6E		Rd		Rs1		-		#am	
31	24	23	20	19	16	15	4	3	0

COM							
# 0x7C		Rd		Rs1		-	
31	24	23	20	19	16	15	0

NEG							
# 0x8C		Rd		Rs1		-	
31	24	23	20	19	16	15	0

ROL							
# 0x9D		Rd		Rs1		-	
31	24	23	20	19	16	15	0

ROR					
# 0xAD		Rd		Rs1	-
31	24	23	20	19	16	15	0

MOV							
# 0xB8		Rd		-		Rs1	-
31	24	23	20	19	16	15	12	11	0

MOVI					
# 0xBA		Rd		-	#data
31	24	23	20	19	16	15	0

MOVF							
# 0xBC		Rd		-		Rs1	-
31	24	23	20	19	16	15	12	11	0

MULT							
# 0x58		Rd		Rs1		Rs2	-
31	24	23	20	19	16	15	12	11	0

MULTI					
# 0x5A		Rd		Rs1	#data
31	24	23	20	19	16	15	0

DIV							
# 0x49		Rd		Rs1		Rs2	-
31	24	23	20	19	16	15	12	11	0

DIVI					
# 0x4B		Rd		Rs1	#data
31	24	23	20	19	16	15	0

SET		
# 0xD7	-	#mask
31 24	23 16	15 0

LOAD				
# 0xD9	Rd	-	Rs1	-
31 24	23 20	19 16	15 12	11 0

LOADI			
# 0xDB	Rd	-	#addr
31 24	23 20	19 16	15 0

STORE				
# 0xC8	-	Rs1	Rs2	-
31 24	23 20	19 16	15 12	11 0

STOREI			
# 0xCA	-	Rs1	#addr
31 24	23 20	19 16	15 0

IN			
# 0xDE	Rd	-	#sel
31 24	23 20	19 2	1 0

OUT				
# 0xC4	#op	#sel	Rs1	-
31 24	23 22	21 20	19 16	15 0

OUTI				
# 0xC6	#op	#sel	-	#data
31 24	23 22	21 20	19 16	15 0

JMP		
# 0xE0	-	#addr
31 24	23 16	15 0

JC			
# 0xC3	# 0x0	-	#addr
31 24	23 20	19 16	15 0

JNC			
# 0xC3	# 0x1	-	#addr
31 24	23 20	19 16	15 0

JO			
# 0xC3	# 0x2	-	#addr
31 24	23 20	19 16	15 0

JNO			
# 0xC3	# 0x3	-	#addr
31 24	23 20	19 16	15 0

JN			
# 0xC3	# 0x4	-	#addr
31 24	23 20	19 16	15 0

JP			
# 0xC3	# 0x5	-	#addr
31 24	23 20	19 16	15 0

JZ			
# 0xC3	# 0x6	-	#addr
31 24	23 20	19 16	15 0

JNZ			
# 0xC3	# 0x7	-	#addr
31 24	23 20	19 16	15 0

JL			
# 0xC3	# 0x8	-	#addr
31 24	23 20	19 16	15 0

JLE			
# 0xC3	# 0xA	-	#addr
31 24	23 20	19 16	15 0

JG			
# 0xC3	# 0xB	-	#addr
31 24	23 20	19 16	15 0

JGE			
# 0xC3	# 0x9	-	#addr
31 24	23 20	19 16	15 0

CALL		
# 0xE1	-	#addr
31 24	23 16	15 0

RET	
# 0xE2	-
31 24	23 0

PUSH			
# 0xF2	-	Rs1	-
31 24	23 20	19 16	15 0

POP		
# 0xF8	Rd	-
31 24	23 20	19 0

PUSHF	
# 0xF1	-
31 24	23 0

POPF	
# 0xF4	-
31 24	23 0

STOP	
# 0xC1	-
31 24	23 0