The NiMi-iSC 16-bit Instruction-Set Architecture

Revision 3

NiMi-iSC 16-bit Instruction Set overview

This instruction set is meant to be a middle ground between overly generalized architectures such as RiSC-16[1] and overly specific architectures such as x86. NiMiiSC-16 is a 16 register, 16-bit computer. The word size is 16 bits. All busses are and memory addresses are word-addresses (i.e., address 0 corresponds to the first two bytes of main memory, address 1 corresponds to the second two bytes of main memory, etc.). In each instruction 2 bits are dedicated to the instruction mode and 3 bits are dedicated to the opcode. Big endian will be used throughout the instruction set. Due to the purpose of this architecture being for learning purposes, therefore the instructions will not be optimized to reduce die size

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Formats:

RR	Mode	Opcode	Reg a	Reg b	0
I	Mode	Immediate			
D	Mode	Opcode	Data		
R	Mode	Opcode	Reg a	0	
N	Mode	Opcode	0		

Instructions

			ALU		
ADD	00	000	Reg a	Reg b	0
SUB	00	001	Reg a	Reg b	0
AND	00	010	Reg a	Reg b	0
OR	00	011	Reg a	Reg b	0
NOT	00	100	Reg a	0	
XOR	00	101	Reg a	Reg b	0
MUL	00	110	Reg a	Reg b	0
DIV	00	111	Reg a	Reg b	0

[1] (Jacob)

Immediate

Imm	01	Immediate (0 to 0x3FFF)	

Conditions

JMP	10	000	0		
EQ	10	001	Reg a	Reg b	0
NEQ	10	010	Reg a	Reg b	0
LES	10	011	Reg a	Reg b	0
LOE	10	100	Reg a	Reg b	0
GRE	10	101	Reg a	Reg b	0
GOE	10	110	Reg a	Reg b	0

MISC

PUSH	11	000	Reg a	0	
POP	11	001	Reg a	0	
CALL	11	001	Reg a	0	
RET	11	011	0		
LDR	11	100	Reg a	Reg b	0
STR	11	101	Reg a	Reg b	0
BITL	11	110	Reg a	Reg b	0
BITR	11	111	Reg a	Reg b	0

Description of instruction operators

Mnemonic	Name and format	Mode (Binary)	Opcode (Binary)	Assembly format	Description
ADD	Add RR-type	00	000	add rA, rB	Add rA with rB and save result in rA
SUB	Sub RR-type	00	001	sub rA, rB	Subtract rA with rB and save result in rA
AND	And RR-type	00	010	and rA, rB	And rA with rB and save result in rA
OR	Or RR-type	00	011	or rA, rB	Or rA with rB and save result in rA
NOT	Not R-type	00	100	not rA	Not rA and save result in rA
XOR	Xor RR-type	00	101	xor rA, rB	Xor rA with rB and save result in rA
MUL	Multiplicate RR-type	00	110	mul rA, rB	Multiplicate rA with rB and save result in rA
DIV	Divide RR-type	00	111	div rA, rB	Divide rA with rB and save result in rA
IMM	Immediate I-type	01	N/A	imm imm	Place immediate value in AX
JMP	Jump N-type	10	000	jmp	Jump to value pointed by GX
EQ	Equal RR-type	10	001	eq rA, rB	If rA == rB jump to value pointed by GX
NEQ	Not equal RR-type	10	010	neq rA, rB	If rA != rB jump to value pointed by GX
LES	Less RR-type	10	011	les rA, rB	If rA < rB jump to value pointed by GX
LOE	Less or equal RR-type	10	100	loe rA, rB	If rA <= rB jump to value pointed by GX
GRE	Greater RR-type	10	101	gre rA, rB	If rA > rB jump to value pointed by GX
GOE	Greater or equal RR-type	10	110	goe rA, rB	If rA >= rB jump to value pointed by GX
PUSH	Push R-type	11	000	push rA	Push rA onto stack
POP	Pop R-type	11	001	pop rA	Pop stack onto rA
CALL	Call R-type	11	010	call rA	Call function pointed by rA
RET	Return N-type	11	011	ret	Return from function
LDR	Load RR-type	11	100	ldr rA, rB	Load value pointed by rB into rA
STR	Store RR-type	11	101	str rA, rB	Store value held in rA into rB
BITL	Bitshift Left RR-type	11	110	bitl rA, rB	Bit shift left rA by rB amount
BITR	Bitshift right RR-type	11	111	bitr rA, rB	Bit shift right rA by rB amount

Registers

All registers are 16 bit.

0) AX *
1) BX
2) CX
3) DX
4) EX
5) FX
6) GX
7) HX
8) IX **
9) JX ***
10) KX
11) LX
12) MX
13) NX
14) Counter
15) I/O ****

* AX: Is always 0 regardless of any write operations to it

** IX: stores the immediate value when Imm is used

*** JX: will be used as a jump address when conditional mode is used

**** I/O register will receive user input when it is placed in an input position, and it will output to user when put in an output register position.