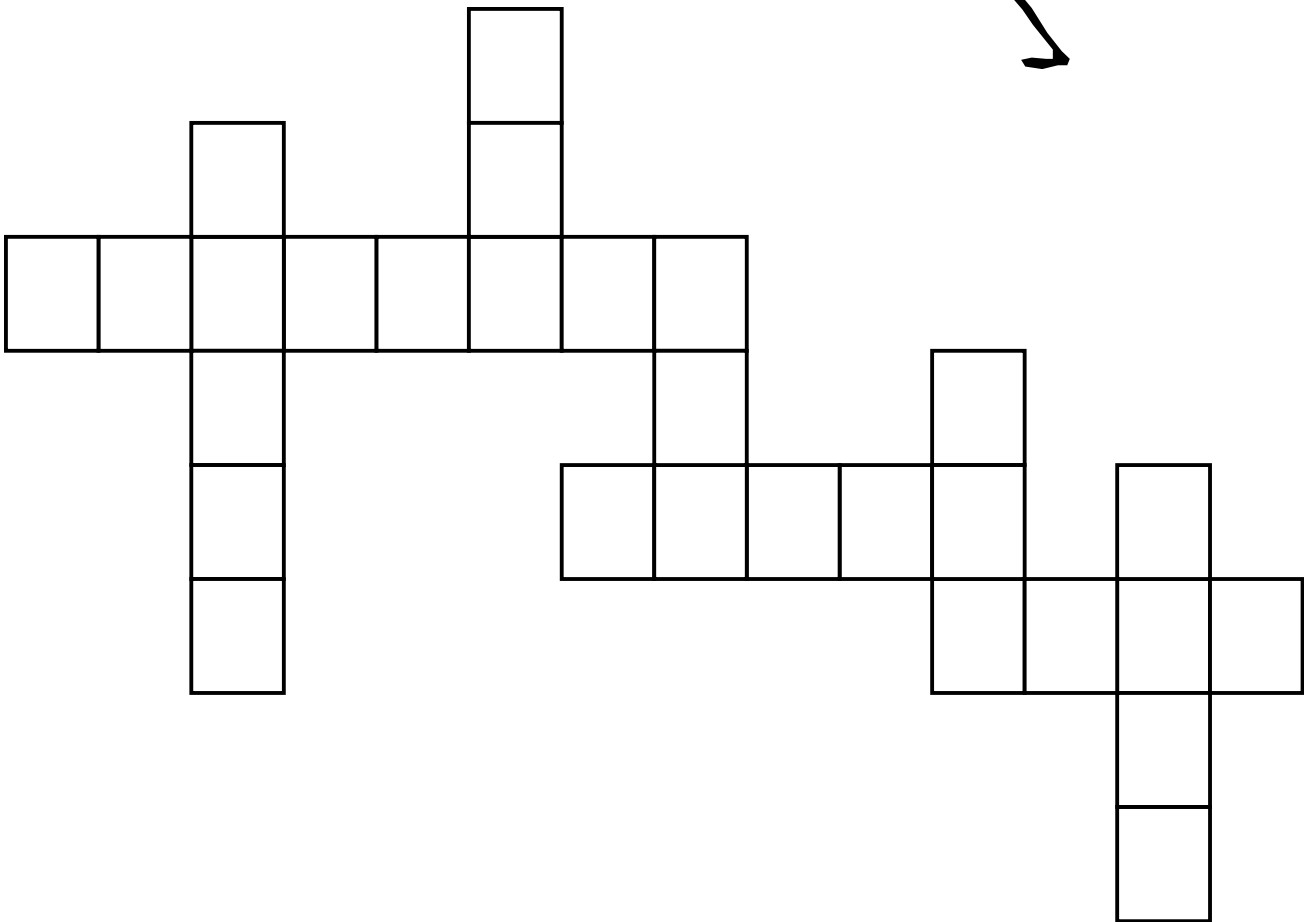


MMIX

A Lesson About MMIX (for beginners)





Criteria for a RISC architecture

Reduced instruction set

“The 256 possible OP codes fall into a dozen or so easily remembered categories”

“...the instruction set is quite convenient”

Arithmetical / logical instructions

Load / Store instructions

Comparisons

Branches & Jumps

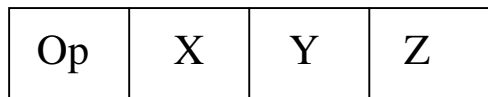
Subroutine Call/ Return

System Calls

...

Horizontal Instruction format

“Each instruction in MMIX has the four-byte form OP X Y Z”



Fast execution of the instructions

Most instructions are executed in one machine cycle.

This is achieved by pipelining.



Criteria for a RISC architecture

Memory access only by Load/Store Instructions

All arithmetical and logical instructions are working with register operands.

Many general purpose registers

*“There are at least 256 general-purpose registers:
\$0, \$1, \$2, ... \$255”*

Simple Addressing mechanism

Indirect addressing is not supported.





Example

MMIX Op Codes

Each instruction in MMIX has the four-byte form OP X Y Z, where OP is one of the following 256 operations:

	0	1	2	3	4	5	6	7	
0x	TRAP	FCMP	FUN	FEQL	FADD	FIX	FSUB	FIXU	0x
	FLOT[I]		FLOTU[I]		SFLOT[I]		SFLOTU[I]		
1x	FMUL	FCMPE	FUNE	FEQLE	FDIV	FSQRT	FREM	FINT	1x
	MUL[I]		MULU[I]		DIV[I]		DIVU[I]		
2x	ADD[I]	ADDU[I]			SUB[I]		SUBU[I]	2x	
	2ADDU[I]		4ADDU[I]		8ADDU[I]		16ADDU[I]		
3x	CMP[I]	CMPU[I]			NEG[I]		NEGU[I]	3x	
	SL[I]		SLU[I]		SR[I]		SRU[I]		
4x	BN[B]	BZ[B]			BP[B]		BOD[B]	4x	
	BNN[B]		BNZ[B]		BNP[B]		BEV[B]		
5x	PBN[B]	PBZ[B]			PBP[B]		PBOD[B]	5x	
	PBNN[B]		PBNZ[B]		PBNP[B]		PBEV[B]		
6x	CSN[I]	CSZ[I]			CSP[I]		CSOD[I]	6x	
	CSNN[I]		CSNZ[I]		CSNP[I]		CSEV[I]		
7x	ZSN[I]	ZSZ[I]			ZSP[I]		ZSOD[I]	7x	
	ZSNN[I]		ZSNZ[I]		ZSNP[I]		ZSEV[I]		
8x	LDB[I]	LDBU[I]			LDO[I]		LDWU[I]	8x	
	LDT[I]		LDTU[I]		LDOU[I]		LDUNC[I]		
9x	LDSF[I]	LDHT[I]			LDST[I]		LDUNC[I]	9x	
	LDVTS[I]		PRELD[I]		PREGO[I]		GO[I]		
Ax	STB[I]	STBU[I]			STO[I]		STWU[I]	Ax	
	STT[I]		STTU[I]		STOU[I]		STUNC[I]		
Bx	STSF[I]	STHT[I]			STCO[I]		STUNC[I]	Bx	
	SYNCD[I]		PREST[I]		SYNCID[I]		PUSHGO[I]		
Cx	OR[I]	ORN[I]			NOR[I]		XOR[I]	Cx	
	AND[I]		ANDN[I]		NAND[I]		NXOR[I]		
Dx	BDIF[I]	WDIF[I]			TDIF[I]		ODIF[I]	Dx	
	MUX[I]		SADD[I]		MOR[I]		MXOR[I]		
Ex	SETH	SETMH	SETML	SETL	INCH	INCMH	INCML	INCL	Ex
	ORH	ORMH	ORML	ORL	ANDNH	ANDNMH	ANDNML	ANDNL	
Fx	JMP[B]		PUSHJ[B]		GETA[B]		PUT[I]		Fx
	POP	RESUME	SAVE	UNSAVE	SYNC	SWYM	GET	TRIP	
	8	9	A	B	C	D	E	F	



Example

LOAD Instruction

LDO Load Octa (8 Bytes)

The contents of a memory location is transferred to a register.
The addresses are aligned: the last 3 Bits are ignored.

LDO \$X,\$Y,\$Z

\$X : destination register
\$Y, \$Z : source address

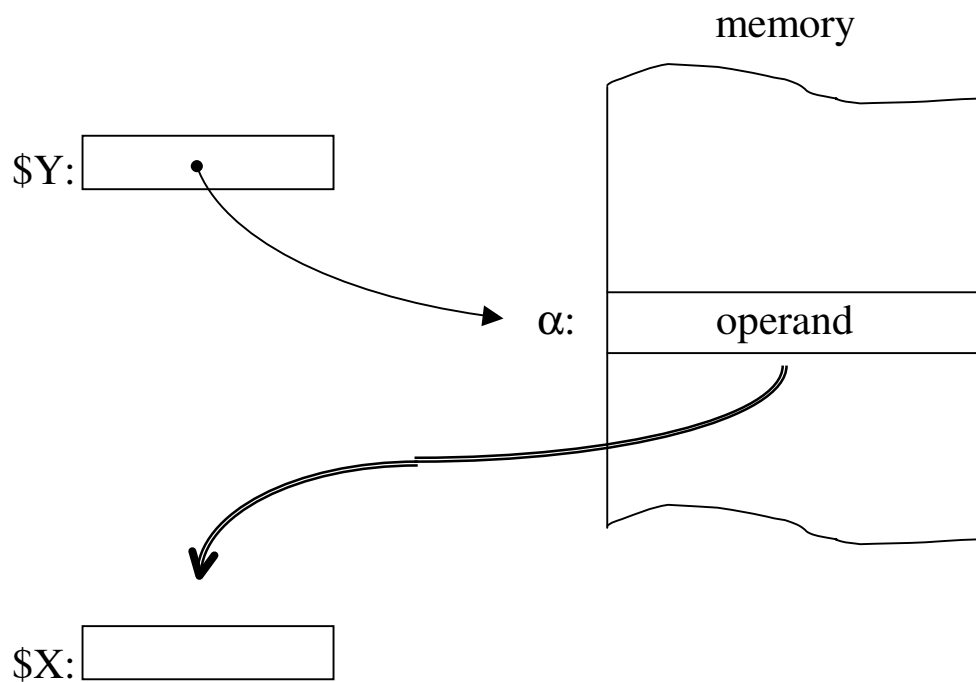
Depending on the format of \$Z the memory address
is computed in one of the following ways.



1. case

LDO \$X,\$Y,0

Register \$Y contains the **absolute address** α of the operand.



Usage

Loading a variable into a register



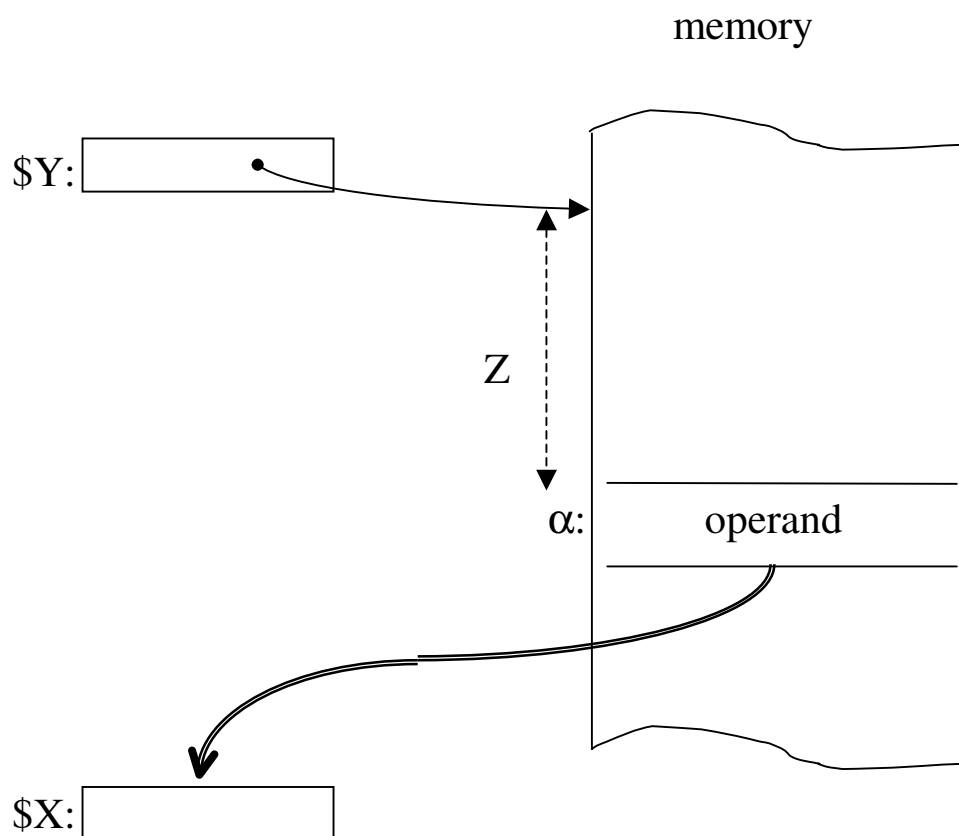
2. case

LDO \$X,\$Y,**Z**

Register \$Y contains the **base address** of the operand,
Z is the (fixed) **offset**.

The address α is computed by

$$\alpha = \$Y + Z$$



Usage

Accessing the elements of a record



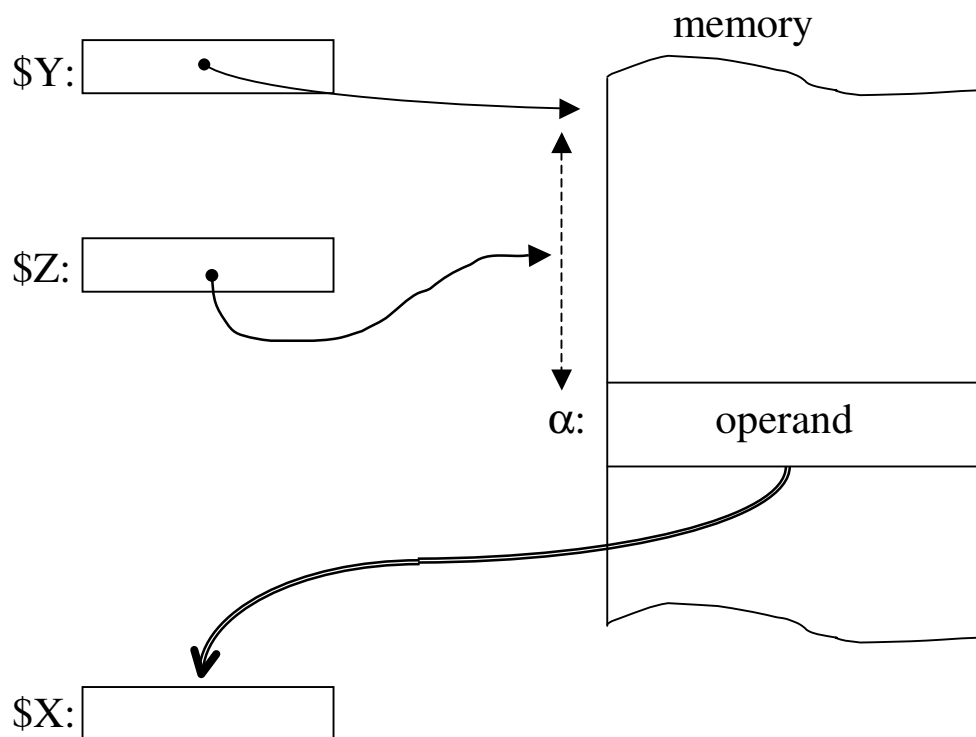
3. case

LDO \$X, \$Y, \$Z

Register \$Y contains the **base address** of the operand,
\$Z contains the (variable) **offset**.

The address α is computed by

$$\alpha = \$Y + \$Z$$



Usage

Addressing the elements of a vector



Example

	LOC	Data_Segment	
A	OCTA	#0001020304050607	
	OCTA	#08090A0B0C0D0E0F	
	LOC #100		
Main	LDA	\$2,A	load base address
	LDO	\$1,\$2,0	case 1: absolute address
	LDO	\$1,\$2,8	case 2: fixed offset
	LDO	\$1,\$2,4	alignment!
	SETL	\$3,0	case 3: variable offset
	LDO	\$1,\$2,\$3	
	ADD	\$3,\$3,8	
	LDO	\$1,\$2,\$3	
	TRAP	0,Halt,0	



* Example MMIX program

Dest	IS	\$1	Destination register
Base	IS	\$2	Base register
Offset	IS	\$3	Offset register

LOC Data_Segment

A OCTA #0001020304050607
OCTA #08090A0B0C0D0E0F

LOC #100

Main LDA Base,A Load base address

LDO Dest,Base,0 case 1: absolute address

LDO Dest,Base,8 case 2: fixed offset
LDO Dest,Base,4 alignment!

SETL Offset,0 case 3 : variable offset
LDO Dest,Base,Offset
ADD Offset,Offset,8
LDO Dest,Base,Offset

TRAP 0,Halt,0



* Example MMIX program

Word	IS	\$1	Destination register
Base	IS	\$2	Base register
Offset	IS	\$3	Offset register

	LOC	Data_Segment	
	BYTE	"without "	
	OCTA	6*8	
	BYTE	"practice "	
	OCTA	0	
	BYTE	"no "	
	OCTA	2*8	
	BYTE	"theory "	
	OCTA	4*8	
FIRST	OCTA	4*8	
	LOC	#100	
Main	LDA	Base,Data_Segment	base
	LDO	Offset,FIRST	relative offset
⇒	LDO	Word,Base,Offset	1. word:
	ADD	Offset,Offset,8	
	LDO	Offset,Base,Offset	
⇒	LDO	Word,Base,Offset	2. word:
	ADD	Offset,Offset,8	
	LDO	Offset,Base,Offset	
⇒	LDO	Word,Base,Offset	3. word:
	ADD	Offset,Offset,8	
	LDO	Offset,Base,Offset	
⇒	LDO	Word,Base,Offset	4. word:
	TRAP	0,Halt,0	