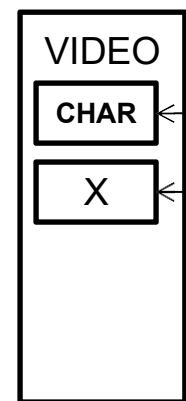
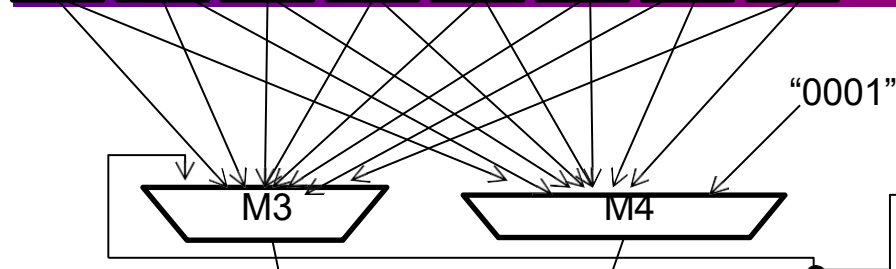
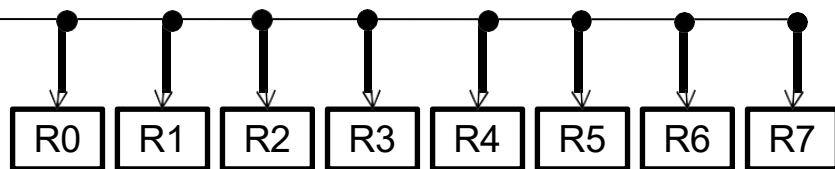

Lab ORG

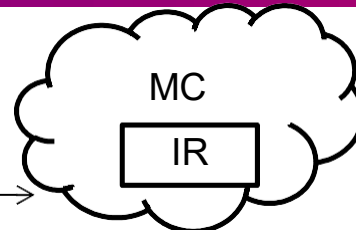
Processador ICMC

Eduardo Simões

Arquitetura

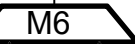


"0001"



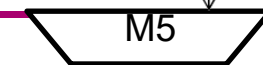
MC

IR



auxFR

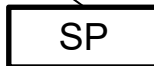
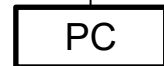
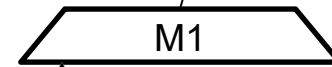
Mem



MEMÓRIA



TECLADO



Conjunto de registradores do uP ICMC

Nome	Qtde	Finalidade
R_n	0-7	Registradores de propósito geral
FR	1	Flag Register
SP	1	Ponteiro da pilha
PC	1	Contador de programa
IR (interno)	1	Registrador de instruções
MAR (interno)	1	Registrador de endereço de memória

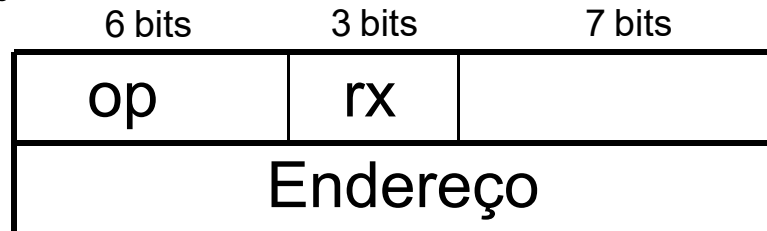
- Arquitetura RISC do tipo Load/Store
- Operações de Reg. para Reg.

Formato de Instrução

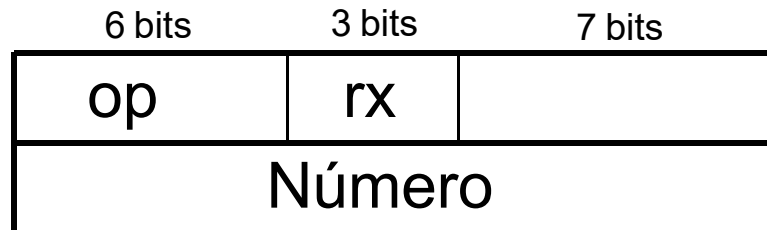
- Manipulação de Dados

- op = opcode
- rx, ry, rz: registradores
- c: uso do bit de carry

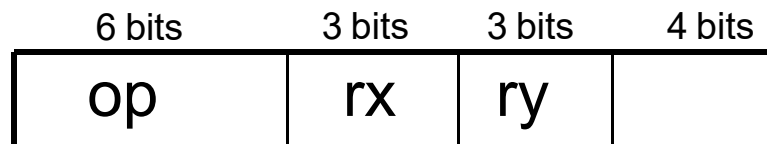
- Direto:



- Imediato:



- Indireto por Registrador



Instruções de manipulação de dados

Direto

STORE END, RX	MEM(END) <- RX	110001 RX xxx xxx x END
LOAD RX, END	RX <- MEM(END)	110000 RX xxx xxx x END

Indireto por Registrador

STOREI RX, RY	MEM(RX) <- RY	111101 RX RY xxx x
LOADI RX, RY	RX <- MEM(RY)	111100 RX RY xxx x

Imediato

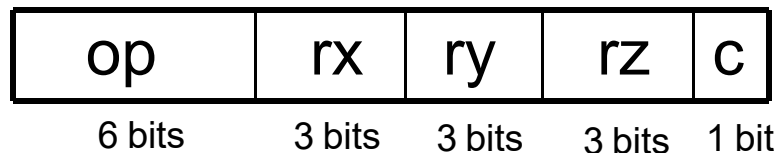
LOADN RX, #NR	RX <- NR	111000 RX xxx xxx x NR
---------------	----------	-----------------------------------

Movimentação

MOV RX, RY	RX <- RY	110011 RX RY xx x0
MOV RX, SP	RX <- SP	110011 RX xxx xx 01
MOV SP, RX	SP <- RX	110011 RX xxx xx 11

Formato de Instrução

- Instruções Lógicas e Aritméticas
 - op = opcode
 - rx, ry, rz: registradores
 - c: uso do bit de carry



Instruções aritméticas

ADD RX, RY, RZ	$RX \leftarrow RY + RZ$	100000 RX RY RZ 0
ADDC RX, RY, RZ	$RX \leftarrow RY + RZ + C$	100000 RX RY RZ 1
SUB RX, RY, RZ	$RX \leftarrow RY - RZ$	100001 RX RY RZ 0
SUBC RX, RY, RZ	$RX \leftarrow RY - RZ + C$	100001 RX RY RZ 1
MULT RX, RY, RZ	$RX \leftarrow RY * RZ$	100010 RX RY RZ 0
MULTC RX, RY, RZ	$RX \leftarrow RY * RZ + C$	100010 RX RY RZ 1
DIV RX, RY, RZ	$RX \leftarrow RY / RZ$	100011 RX RY RZ 0
DIVC RX, RY, RZ	$RX \leftarrow RY / RZ + C$	100011 RX RY RZ 1
INC RX	$RX++$	100100 RX 0 xxx xxx
DEC RX	$RX--$	100100 RX 1 xxx xxx
MOD RX, RY, RZ	$RX \leftarrow RY \text{ MOD } RZ$	100101 RX RY RZ x

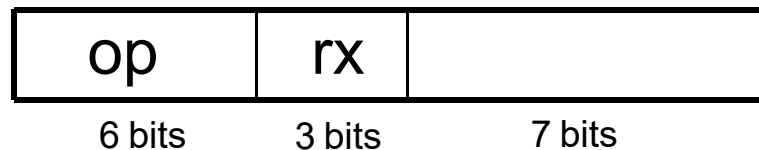
Instruções lógicas

AND RX, RY, RZ	RX<-RY AND RZ	010010 RX RY RZ x
OR RX, RY, RZ	RX<-RY OR RZ	010011 RX RY RZ x
XOR RX, RY, RZ	RX<-RY XOR RZ	010100 RX RY RZ x
NOT RX, RY	RX<-NOT(RY)	010101 RX RY xxx x
ROTL RX,n	ROTATE TO LEFT	010000 RX 10x nnn n
ROTR RX,n	ROTATE TO RIGHT	010000 RX 11x nnn n
SHIFTL0 RX,n	SHIFT TO LEFT (FILL 0)	010000 RX 000 nnn n
SHIFTL1 RX,n	SHIFT TO LEFT (FILL 1)	010000 RX 001 nnn n
SHIFTR0 RX,n	SHIFT TO RIGHT (FILL 0)	010000 RX 010 nnn n
SHIFTR1 RX,n	SHIFT TO RIGHT (FILL 1)	010000 RX 011 nnn n
CMP RX, RY	FR<-COND	010110 RX RY xxx x

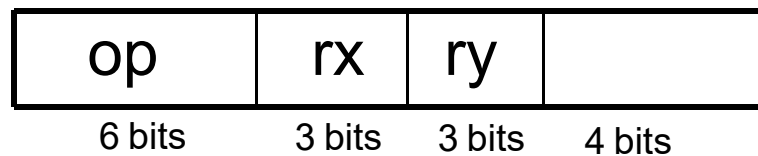
Formato de Instrução

- Instruções de entrada e saída

- Input



- Output



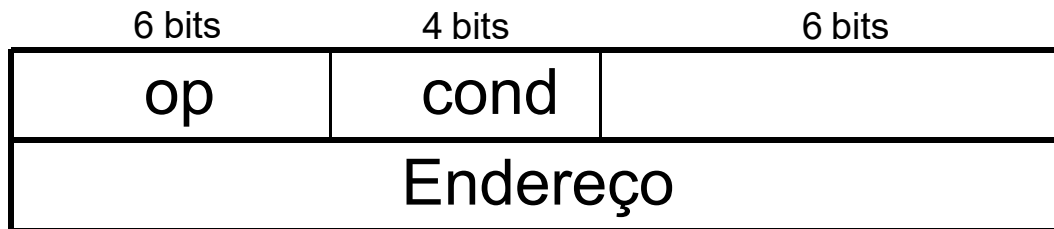
Instruções de entrada e saída

INCHAR RX RX<-"00000000"&key 110101 | RX | xxx | xxx | x

OUTCHAR RX, RY VIDEO(RY)<-CHAR(RX) 110010 | RX | RY | xxx | x

Formato de Instrução

- Controle de desvio



Instruções de salto (todas com END)

Salto se condição verdadeira para o END

JMP END	PC<-END	unconditional	000010 0000 x xxxxx
		END	
JEQ END	PC<-END	Equal	000010 0001 x xxxxx
JNE END	PC<- END	NotEqual	000010 0010 x xxxxx
JZ END	PC<- END	Zero	000010 0011 x xxxxx
JNZ END	PC<- END	NotZero	000010 0100 x xxxxx
JC END	PC<- END	Carry	000010 0101 x xxxxx
JNC END	PC<- END	NotCarry	000010 0110 x xxxxx
JGR END	PC<- END	GReater	000010 0111 x xxxxx
JLE END	PC<- END	LEsser	000010 1000 x xxxxx
JEG END	PC<- END	EqualorGreater	000010 1001 x xxxxx
JEL END	PC<- END	EqualorLesser	000010 1010 x xxxxx
JOV END	PC<- END	Overflow (ULA)	000010 1011 x xxxxx
JNOV END	PC<- END	NotOverflow	000010 1100 x xxxxx
JN END	PC<-END	Negative (ULA)	000010 1101 x xxxxx
JDZ END	PC<-END	DivbyZero	000010 1110 x xxxxx

Novas Instruções de salto (todas com END)

Salto se condição verdadeira para o END

JNN END	PC<-END	Not Negative	000010 1111 x xxxxx
JGZ END	PC<- END	Greater Zero	001011 0000 x xxxxx
JGEZ END	PC<- END	Greater Equal Zero	001011 0001 x xxxxx
JLZ END	PC<- END	Less Zero	001011 0010 x xxxxx
JLEZ END	PC<- END	Less Equal Zero	001011 0011 x xxxxx

Instruções de chamada (todas com END)

Chama procedimento se condição verdadeira

CALL END	MEM(SP)<-PC	Unconditional	000011 0000 x xxxxx
	PC<-END		END
	SP--		
CEQ END	idem	EQual	000011 0001 x xxxxx
CNE END	idem	NotEqual	000011 0010 x xxxxx
CZ END	idem	Zero	000011 0011 x xxxxx
CNZ END	idem	NotZero	000011 0100 x xxxxx
CC END	idem	Carry	000011 0101 x xxxxx
CNC END	idem	NotCarry	000011 0110 x xxxxx
CGR END	idem	GReater	000011 0111 x xxxxx
CLE END	idem	LEsser	000011 1000 x xxxxx
CEG END	idem	EqualorGreater	000011 1001 x xxxxx
CEL END	idem	EqualorLesser	000011 1010 x xxxxx
COV END	idem	Overflow (ULA)	000011 1011 x xxxxx
CNOV END	idem	NotOverflow	000011 1100 x xxxxx
CN END	idem	Negative (ULA)	000011 1101 x xxxxx
CDZ END	idem	DivbyZero	000011 1110 x xxxxx

Novas Instruções de chamada (todas com END)

Chama procedimento se condição verdadeira

CNN END	MEM(SP)<-PC PC<-END SP--	Not Negative	000011 1111 x xxxxx
CGZ END	idem	Greater Zero	001100 0000 x xxxxx
CGEZ END	idem	Greater Equal Zero	001100 0001 x xxxxx
CLZ END	idem	Less Zero	001100 0010 x xxxxx
CLEZ END	idem	Less Equal Zero	001100 0011 x xxxxx

Instrução de retorno

RTS	SP++	000100 xxxx x xxxxx
	PC<=MEM(SP)	
	PC++	

Obs.: - Não esquecer de incrementar o PC pois foi guardado na pilha ainda apontando para o END no CALL.

Formato de Instrução

- Pilha

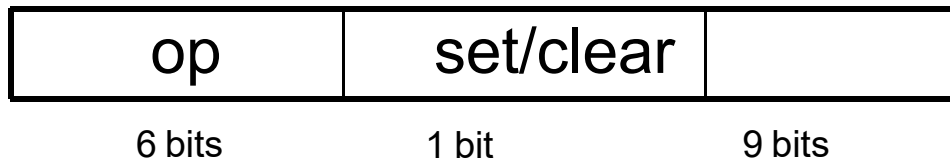


Instruções de pilha

PUSH	RX	MEM(SP) <- RX SP--	000101 RX 0 xxxxxx
PUSH	FR	MEM(SP) <- FR SP--	000101 xxx 1 xxxxxx
POP RX		SP++ MEM(SP) -> RX	000110 RX 0 xxxxxx
POP FR		SP++ MEM(SP) -> FR	000110 xxx 1 xxxxxx

Formato de Instrução

- Controle



Instruções de controle

CLEARC	C<-0	001000 0 xxxxxxxxx
SETC	C<-1	001000 1 xxxxxxxxx
HALT	STOP EXECUTION	001111 x xxxxxxxxx
NOOP	NO OPERATION	000000 x xxxxxxxxx
BREAKP	Insert Break Point	001110 x xxxxxxxxx