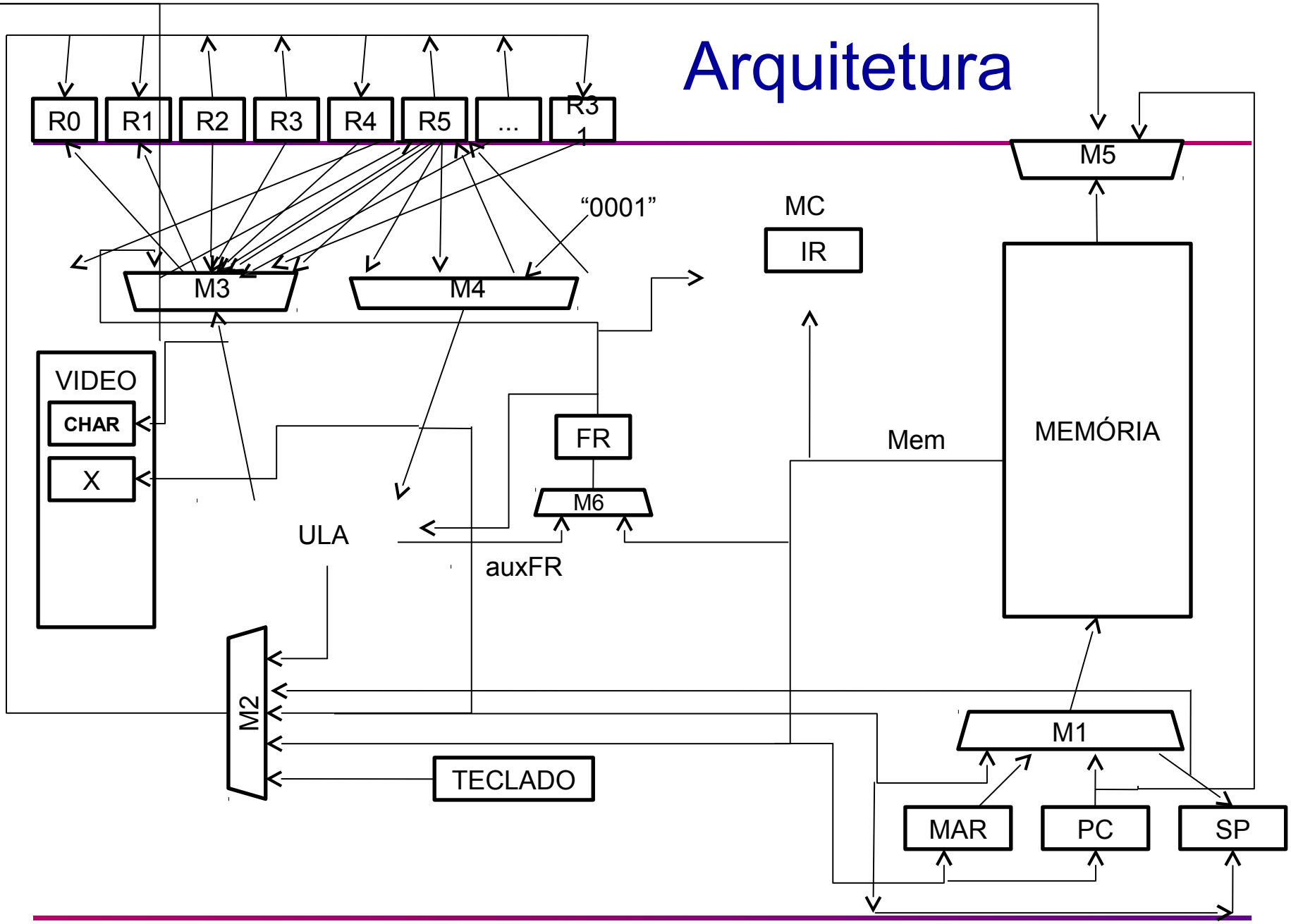
Lab ORG

Processador AP9 em Verilog 32bits

Grupo 1: Anderson, Leandro, Thayton – Intruçoes em VERDE
Grupo 2: Douglas, Antonio, Fabiano – Instrucoes em AMARELO
Grupo 3: Gustavo, Eric, Cassia – Instrucoes em AZUL
Grupo 4: Nelson, Rennan, Wendel – Instrucoes em VERMELHO
Grupo 5: Leandro Marega, Diogo – Instrucoes em PRETO

Arquitetura



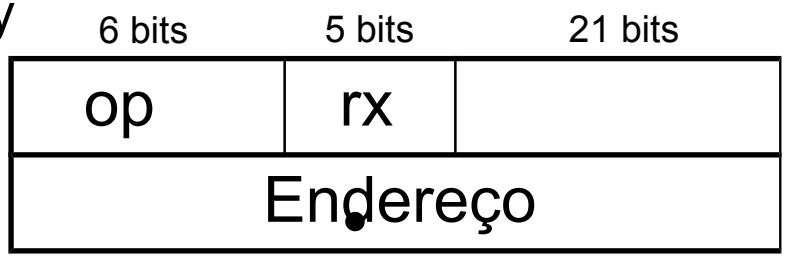
Conjunto de registradores do uP ICMC

Nome	Qtde	Finalidade
R_n	0-31	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

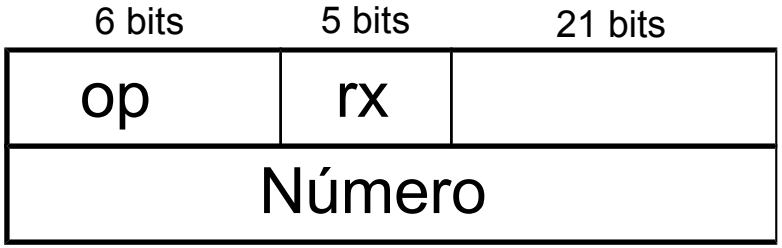
- Arquitetura RISC do tipo Load/Store
- Operações de Reg. para Reg.

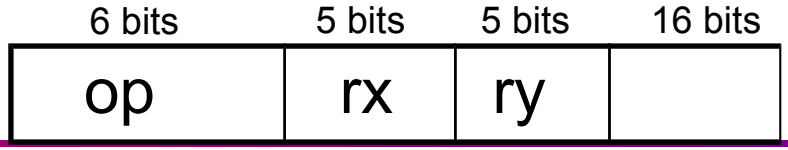
Formato de Instrução

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



    - Imediato:





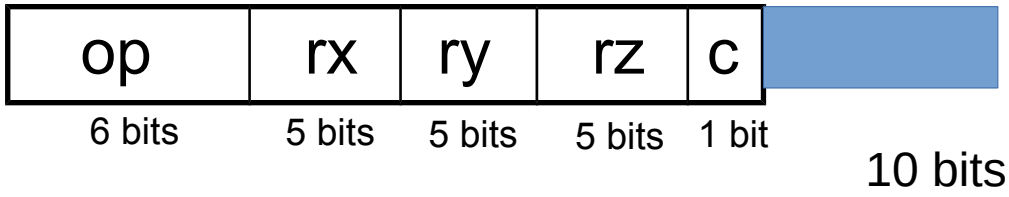
    - Indexado
      Registrador

Grupo 2 manipulação de dados

Direto		
STORE32 END, RX	MEM(END) <- RX END	010111 | RX | xxx | xxx | x
LOAD32 RX, END	RX <- MEM(END) END	011000 | RX | xxx | xxx | x
Indexado		
STOREI32 RX, RY	MEM(RX) <- RY	011001 | RX | RY | xxx | x
LOADI32 RX, RY	RX <- MEM(RY)	011010 | RX | RY | xxx | x
Imediato		
LOADN32 RX, #NR	RX <- NR NR	011011 | RX | xxx | xxx | x
Movimentação		
MOV32 RX, RY	RX <- RY	011100 | RX | RY | xx | x0
MOV32 RX, SP	RX <- SP	011100 | RX | xxx | xx | 01
MOV32 SP, RX	SP <- RX	011100 | RX | xxx | xx | 11

Formato de Instrução

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



Grupo 5 - Instruções aritméticas

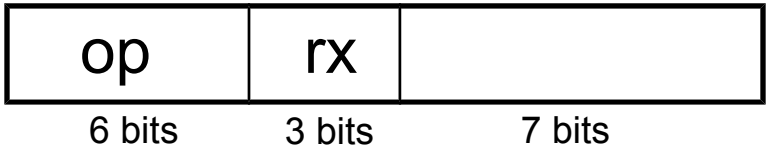
ADD32 RX, RY, RZ	RX<-RY+RZ	011101 | RX | RY | RZ | 0
ADDC32 RX, RY, RZ	RX<-RY+RZ+C	011101 | RX | RY | RZ | 1
SUB32 RX, RY, RZ	RX<-RY-RZ	011110 | RX | RY | RZ | 0
SUBC32 RX, RY, RZ	RX<-RY-RZ+C	011110 | RX | RY | RZ | 1
MUL32 RX, RY, RZ	RX<-RY*RZ	011111 | RX | RY | RZ | 0
DIV32 RX, RY, RZ	RX<-RY/RZ	110100 | RX | RY | RZ | 0
INC32 RX	RX++	110110 | RX | 0 |xxx|xxx |
DEC32 RX	RX--	110110 | RX | 1 |xxx|xxx |
MOD32 RX, RY, RZ	RX<-RY MOD RZ	110111 | RX | RY | RZ | x

Grupo 4 - Instruções lógicas

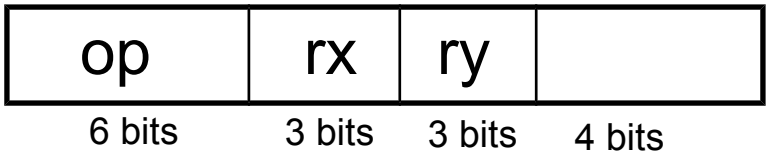
AND32 RX, RY, RZ	RX<-RY AND RZ	101000 | RX | RY | RZ | x
OR32 RX, RY, RZ	RX<-RY OR RZ	101001 | RX | RY | RZ | x
XOR32 RX, RY, RZ	RX<-RY XOR RZ	101010 | RX | RY | RZ | x
NOT32 RX, RY	RX<-NOT(RY)	101011 | RX | RY | xxx | x
ROTL32 RX,n	ROTATE TO LEFT	101100 | RX | 10x | nnn | n
ROTR32 RX,n	ROTATE TO RIGHT	101100 | RX | 11x | nnn | n
SHIFTL32 RX,n	SHIFT TO LEFT (FILL 0)	101100 | RX | 000 | nnn | n
SHIFTR32 RX,n	SHIFT TO RIGHT (FILL 0)	101100 | RX | 010 | nnn | n
CMP32 RX, RY	FR<-COND	101101 | RX | RY | xxx | x

Formato de Instrução (r0 - r7)

- Instruções de entrada e saída
  - Input



  - Output

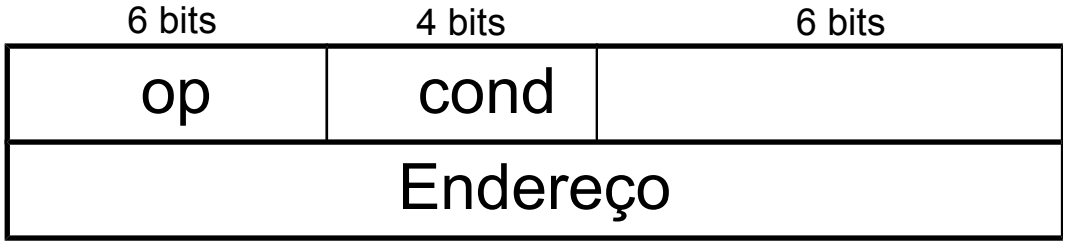


Instruções de entrada e saída

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 END	000010 | 0000 | x | xxxxx
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

Instruções de chamada (todas com END)

Chama procedimento se condição verdadeira

CALL END	MEM(SP)<-PC PC<-END SP--	Unconditional	000011 | 0000 | x | xxxxx END
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

Instrução de retorno

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

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

Formato de Instrução

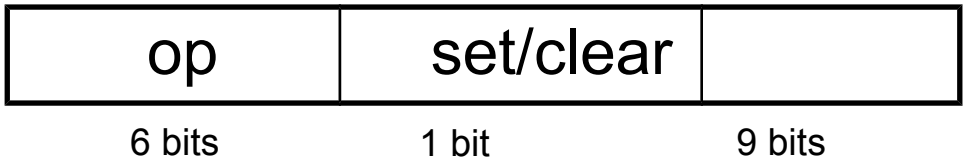
- Pilha

Grupo 3 - Instruções de pilha

PUSH32 RX	MEM(SP) <- RX SP--	001010 | RX | 0 | xxxxxx
PUSH32 FR	MEM(SP) <- FR SP--	001010 | xxx | 1 | xxxxxx
POP32 RX	SP++ MEM(SP) -> RX	001011 | RX | 0 | xxxxxx
POP32 FR	SP++ MEM(SP) -> FR	001011 | 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