| Instruções da Máquina Nativa | | | Instruções da Máquina Virtual | | | | | |
|------------------------------------------------------------------------------------------|----------------------------------|-----------------------------------------|-----------------------------------------|-------------------------|-----------------------------------------|----------|----------------|-------------------------------------------------------|
| Transferência Memória-Registo (<i>Load</i>) Cálculo c/ Inteiros: Operações Aritméticas | | | | Salto Relativo (Branch) | | | | |
| lb | Rdst,addr | add | Rdst, Rsrc1, Rsrc2 | | | b | Label | |
| lbu | Rdst,addr | addi | Rdst, Rsrc, Imm | | | beqz | Rsrc, Label | |
| lw | Rdst,addr | addiu | Rdst, Rsrc, Imm | | | bge | Rsrc, Src, La | bel |
| | | addu | Rdst, Rsrc1, Rsrc2 | | | bgeu | Rsrc, Src, La | bel |
| | | sub | Rdst, Rsrc1, Rsrc2 | | | bgt | Rsrc, Src, La | pel |
| Transfer | ência Registo-Memória (Store) | subu | Rdst, Rsrc1, Rsrc2 | | | bgtu | Rsrc, Src, La | pel |
| sb | Rsrc,addr | | | | | ble | Rsrc, Src, La | pel |
| SW | Rsrc,addr | | | Transfe | rência Registo-Registo (Move) | bleu | Rsrc, Src, La | pel |
| | | | | move | Rdst, Rsrc | blt | Rsrc, Src, La | pel |
| | | | | | | bltu | Rsrc, Src, La | bel |
| | | Cálculo | c/ Inteiros: Op. Lógicas <i>Bitwise</i> | Manipu | lação de Const. (Load Imm/sym) | bnez | Rsrc,Label | |
| | | and | Rdst, Rsrc1, Rsrc2 | la | Rdst,sym | | | |
| | | andi | Rdst, Rsrc, Imm | li | Rdst, IMM | | | |
| | | nor | Rdst, Rsrc1, Rsrc2 | | | | Arguitet | ura de Computadores II |
| | | or | Rdst, Rsrc1, Rsrc2 | | | | -,- | DETI-UA |
| | | ori | Rdst, Rsrc, Imm | | | | | |
| | | xor | Rdst, Rsrc1, Rsrc2 | Cálculo | c/ Inteiros: Op. Aritméticas | Ta | abela resumida | das instruções da arquitetura MIPS |
| | | xori | Rdst, Rsrc, Imm | abs | Rdst, Rsrc | | | • |
| | | Cálculo o | c/ Inteiros: Operações de Shift | div | Rdst, Rsrc, Src | | | |
| | | sll | Rdst, Rsrc1, Imm5 | divu | Rdst, Rsrc, Src | | | |
| Manipula | ação de Const. (Load Immediate) | sllv | Rdst, Rsrc1, Rsrc2 | mul | Rdst, Rsrc, Src | | | |
| lui | Rdst,Imm | sra | Rdst, Rsrc1, Imm5 | mulo | Rdst, Rsrc, Src | | Tabela I: Reg | gistos do MIPS e convenção de uso |
| | | srav | Rdst, Rsrc1, Rsrc2 | mulou | Rdst, Rsrc, Src | Nome L | | Uso Convencionado |
| Instruçõe | es de Comparação | srl | Rdst, Rsrc1, Imm5 | rem | Rdst, Rsrc, Src | \$zero | \$0 | Constante 0 |
| slt | Rdst,Rsrc1,Rsrc2 | srlv | Rdst, Rsrc1, Rsrc2 | remu | Rdst, Rsrc, Src | \$at | \$1 | Reservado pelo assemblador |
| sltu | Rdst, Rsrc1, Rsrc2 | | | neg | Rdst, Rsrc | \$v0\$v1 | \$2\$3 | Cálculo de expressões e valor de retorno das funções. |
| slti | Rdst, Rsrc, Imm | | | negu | Rdst, Rsrc | \$a0\$a3 | | Primeiros 4 parâmetros das funções |
| sltiu | Rdst, Rsrc, Imm | | | | | \$t0\$t7 | \$8\$15 | Geral (não são preservados pelas funções) |
| | | | | Cálculo | c/ Inteiros: Op. Lógicas Bitwise | \$s0\$s7 | \$16\$23 | Geral (não podem ser alterados pelas funções) |
| Salto Rel | ativo (Branch) e Absoluto (Jump) | | | not | Rdst, Rsrc | \$t8\$t9 | \$24\$25 | Geral (não são preservados pelas funções) |
| beq | Rsrc1, Rsrc2, Label | | | | | \$k0\$k1 | | Reservado pelo kernel do S.O. |
| bgez | Rsrc,Label | 1 | | Cálculo | c/ Inteiros: Operações de <i>Rotate</i> | \$gp | \$28 | Ponteiro para área global (Global Pointer) |
| bgtz | Rsrc,Label | 1 | | rol | Rdst, Rsrc, Src | \$sp | \$29 | Stack Pointer |
| blez | Rsrc,Label | 1 | | ror | Rdst, Rsrc, Src | \$fp | \$30 | Frame Pointer |
| bltz | Rsrc, Label | 1 | | | | \$ra | \$31 | Endereço de retorno das funções (Return Address) |
| bne | Rsrc1, Rsrc2, Label | | | | | | • | - |
| j | Label | 1 | | | | | | |
| jal | Label | 1 | | | | | | |
| jalr | Rsrc | 1 | | | | | | |
| jr | Rsrc | | | | | | | |
| | | Manipulação de Excepções e <i>Traps</i> | | 1 | | | | |
| | | break | n | 1 | | | | |
| | | nop | | | | | | |
| | | eret | | | | | | JLA, MBC, Fev / 2025 |
| | | syscal | 1 | | | | | · · · · · · · · · · · · · · · · · · |
| L | | 1 - | | 1 | | | | |

| Imm | Valor imediato (constante) de 16 bits | addr | Endereço na forma $Imm(Rsrc) = (Rsrc) + Imm$ |
|-----------|-----------------------------------------|-----------------------|----------------------------------------------------|
| IMM | Valor imediato de 32 bits | B _k (Rsrc) | Byte índice k de Rsrc |
| Rsrc(1,2) | Registo fonte (1 ou 2) | FPdst | Registo destino do coprocessador aritmético |
| (Rsrc) | Conteudo de Rsrc | FPsrc(1,2) | Registo fonte do coprocessador aritmético (1 ou 2) |
| Rdst | Registo destino | Cz | Coprocessador nº z |
| CReg | Registo do Coprocessador C _z | Src | Rsrc ou IMM |
| sym | Endereço do símbolo (label) sym | Imm5 | Valor imediato (constante) de 5 bits |

| Directivas | Descrição | | | |
|--------------------------------|------------------------------------------------------------------------------------------------------------|--|--|--|
| Para controlo dos Segmento | os · | | | |
| .data [address] | Coloca os próximos items no segmento de dados do utilizador (opcionalmente a partir de address) | | | |
| .text [address] | Coloca os próximos items no segmento de código do utilizador (opcionalmente a partir de <i>address</i>). | | | |
| .kdata [address] | Coloca os próximos items no segmento de dados do kernel (opcionalmente a partir de address). | | | |
| .ktext [address] | Coloca os próximos items no segmento de código do kernel (opcionalmente a partir de address). | | | |
| Para criação de constantes | e variáveis em memória: | | | |
| .ascii str | Armazena uma string em memória sem lhe acrescentar o terminador '\0'. | | | |
| .asciiz str | Armazena uma string em memória acrescentando-lhe o terminador '\0'. | | | |
| .byte b_1 ,, b_n | Armazena as grandezas de 8 bits b ₁ ,, b _n em sucessivos bytes de memória. | | | |
| .word w_1 ,, w_n | Armazena as grandezas de 32 bits w ₁ ,, w _n em sucessivas palavras de memória. | | | |
| .float f_1 ,, f_n | Armazena f_1 ,, f_n em vírgula flutuante, precisão simples (32 bits) no seg. de dados. | | | |
| .double d_1 ,, d_n | Armazena d ₁ ,, d _n em vírgula flutuante, precisão dupla (64 bits) no seg. de dados. | | | |
| .space n | Reserva n bytes no segmento de dados, sem inicializar | | | |
| .equ name, val | Atribui ao símbolo "name" o valor "val" | | | |
| Para controlo do alinhamer | ato: | | | |
| .align n | Alinha o próximo item num endereço múltiplo de 2 ⁿ . | | | |
| Para referências externas: | • | | | |
| .globl sym | Declara que o símbolo sym é global e pode ser referenciado em outros ficheiros. | | | |
| .extern sym size | Declara que o item associado a sym ocupa size bytes e é um símbolo global. | | | |
| .include <filename></filename> | Inclui o ficheiro especificado no campo "filename" | | | |

| Tabela IV: System Calls da placa DETPIC32 | | | | |
|---------------------------------------------------------------|------|-----------------------|---------|--|
| Protótipo equivalent em C | \$v0 | Parâmetros de entrada | Retorno | |
| char inkey(void) | 1 | | \$v0 | |
| char getChar(void) | 2 | | \$v0 | |
| void putChar(char ch) | 3 | \$a0 = character | | |
| unsigned int readInt(unsigned int base) | 4 | a0 = base | \$v0 | |
| int readInt10(void) | 5 | | \$v0 | |
| <pre>void printInt(unsigned int val, unsigned int base)</pre> | 6 | a0 = val, a1 = base | | |
| void printInt10(int val) | 7 | \$a0 | | |
| <pre>void printStr(char *str)</pre> | 8 | a0 = str | | |
| <pre>void readStr(char *buffer, unsigned int nc)</pre> | 9 | a0 = buffer, a1 = nc | | |
| void exit(int code) | 10 | a0 = exit code | | |
| <pre>unsigned int readCoreTimer(void)</pre> | 11 | | \$v0 | |
| void resetCoreTimer(void) | 12 | | | |

printInt(), "base": **16 lsbits** – [2.. 16], **16 msbits** – número de caracteres com que o resultado é apresentado (o valor por omissão é 0, i.e. sem formatação)

| Tabela VI: Registos do CP0 do MIPS | | | | |
|----------------------------------------------------|-----------|----------------------------------------------------|--|--|
| Nome Lógico | Nome Real | Conteúdo | | |
| \$BadVAddr | \$8 | Endereço de memória inválido que causou a excepção | | |
| \$Status | \$12 | Interrupt mask & Enable bits | | |
| \$Cause | \$13 | Tipo de excepção e interrupt bits | | |
| \$EPC | \$14 | Endereço da instrução que causou a excepção | | |
| Tabela VII: Valores dos bits [52] do registo Cause | | | | |
| Valor | Nome | Significado | | |
| 0 | INT | External Interrupt | | |
| 4 | ADDRL | Add error exception (load or store) | | |
| 5 | ADDRS | Add error exception (fetch) | | |
| 6 | IBUS | Bus error on instruction fetch | | |
| 7 | DBUS | Bus error on data load or store | | |
| 8 | SYSCALL | Syscall exception | | |
| 9 | BKPT | Break point exception | | |
| 10 | RI | Reserved instruction exception | | |
| 12 | OVF | Overflow exception | | |