**MIPS R200/R3000**

Contents

[1. Generalitati 2](#_Toc52875694)

[2. Registrii procesor MIPS2000 3](#_Toc52875695)

[3. Moduri de adresare 4](#_Toc52875696)

[4. Asamblor 5](#_Toc52875697)

[4.1. Sintaxa 5](#_Toc52875698)

[4.2. Directive de asamblare: 5](#_Toc52875699)

[5. Set de instructiuni 6](#_Toc52875700)

[5.1. Instructiuni ALU (aritmetico-logice): 6](#_Toc52875701)

[5.2. Instructiuni cu referire la memorie + manipulare constante 7](#_Toc52875702)

[5.3. Instructiuni de intreruperi si exceptii 8](#_Toc52875703)

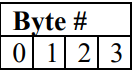
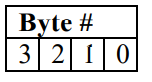
[5.4. Instructiuni de salt si ramificatie 9](#_Toc52875704)

[5.5. Instructiuni de transfer 10](#_Toc52875705)

[5.6. Instructiuni in virgula mobila 11](#_Toc52875706)

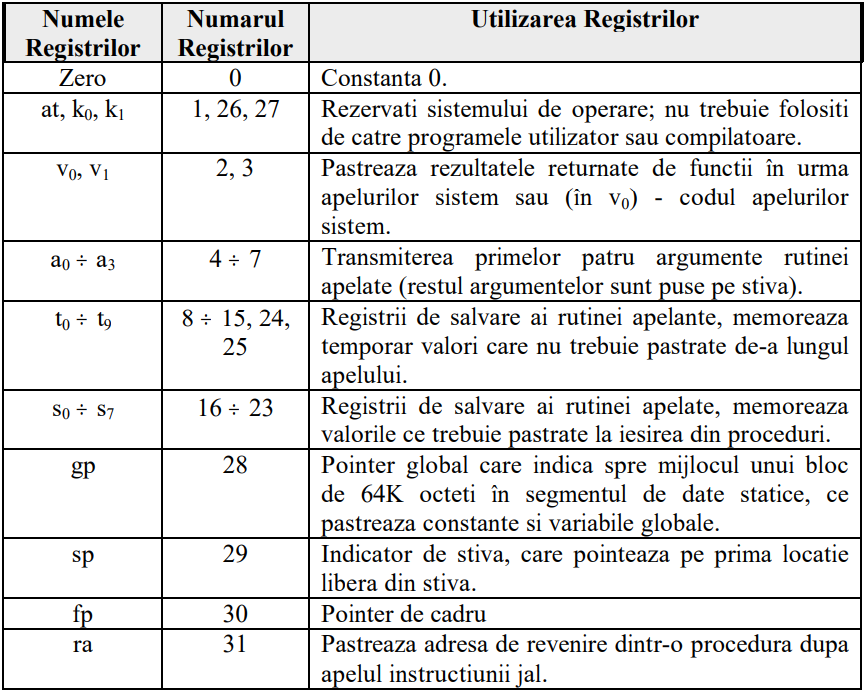
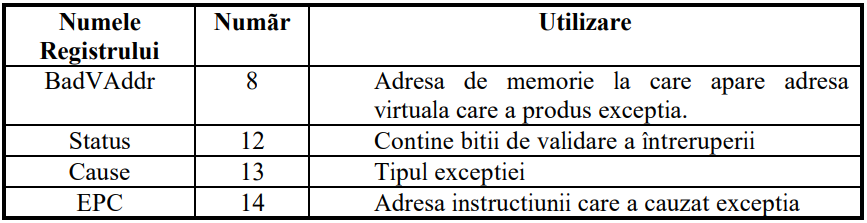
# Generalitati

* Procesor **RISC**, 32 de registrii de uz general
* Unitate de lucru cu intregi + coprocesoare pentru numere in flotant
* Branch/load aduc intarzieri. MIPS are propriul asamblor care contine un set de instructiuni mai bogat decat cel implementat in hardware, astfel el rearanjaza instructiunile astfel incat sa inlature **delay slot-urile** introduse de B si L.
* Simulatorul SPIM/SAL – de 100 de ori mai lent decat un microprocessor real, insa pot detecta mai bine erori, pot fi modificate adaugand instructiuni noi, cu ajutorul lor pot fi colectate date.
* SPIM nu simuleaza memoria cache/latenta memoriei si nu reflecta intarzierile datorate operatiilor in virgula mobila sau a instr div/mul.
* Simulatorul SPIM nu inlatura delay slot-urile astfel instructiunile care se executa sunt identice cu instructiunile programului sursa (nu sunt rearanjate de catre un asamblor)
* Se opereaza atat cu *big-endian* cat si *little-endian* (in functie de masina pe care se opereaza)



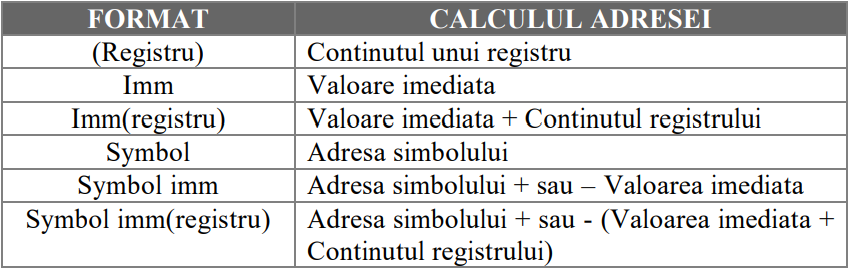
*Big endian Little endian*

# Registrii procesor MIPS2000

* 32 de registrii de uz general, intre 0 – 31, notati $n
* Registrul $0 este cablat la valoarea 0
* registrii coprocesorului 0 (pentru tratare exceptii si intreruperi)

# Moduri de adresare

* **Pentru instructiunile de calcul** (aritmetico-logice) se opereaza doar cu valorile din registre; Procesoarele RISC au un singur mod de adresare a memoriei, **modul indexat**, in care la valoarea din registrul Rx se aduna constanta c;
* Pentru instructiunile Load/Store masinile virtuale asigura alte moduri de adresare:



* Se opereaza cu date aliniate (adresa sa de memorie este un multiplu a marimii sale in octeti); totusi se asigura si existenta unor instructiuni pentru manipularea datelor nealineate (lwl, lwr, swl, swr).

# Asamblor

## Sintaxa

* **Comentariile** incep cu simbolul **#** (#comment)
* **Etichetele** sunt declarate prin asezarea la inceputul liniei urmate de simbolul “**:**” (main: )
* **Identificatorii** sunt cuvinte precedate de simbolul “**.**” sau “**\_**” (.globl, .text, .data)
* Numerele sunt implicit in baza 10. Pentru baza 16, folosim **0x**;
* Sirurile de caractere se pun intre ghilimele; exista caractere speciale precum “\n”, “\t”, “\””

## Directive de asamblare:

* .**align** n – aliniaza datele intr-un domeniu de 2n octeti
* .**asciiz** str – memoreaza sirul str in memorie impreuna cu terminatorul null
* .**byte** b1, … , bn – memoreaza n valori in octeti succesivi in memorie
* .**double** d1, … , dn – numere flotante dubla precizie
* .**float** f1, …, fn – numere flotante simpla precizie
* .**half** h1, … , hn – memoreaza n locatii de 16 biti (halfword) in memorie
* .**word** w1, …, wn – memoreaza cuvinte de 32 de biti in locatii de memorii successive
* .**data** <addr> - numerele declarate ulterior sunt memorate in segmentul de date (incepand cu adresa addr)
* .**text** <addr> - memoreaza date in segmentul de text (incepand cu adresa addr)
* .**globl** sym – declara simbolul *sym* global, astfel poate fi folosit in alte fisiere
* .**kdata** <addr> - date successive memorate in segmentul de date al kernel-ului (incepand cu adresa addr)
* .**ktext** <addr> - date successive memorate in segmentul de text al kernel-ului

# Set de instructiuni

## Instructiuni ALU (aritmetico-logice):

|  |  |
| --- | --- |
| **Op Dest, Src1, (Src2)** | **Descriere** |
| **Adunare** | |
| **add** Rdest, Rsrc1, Src2 | Adunare cu depasire (overflow) |
| **addi** Rdest, Rsrc1, Imm | Adunare imediata cu depasire (overflow) |
| **addu** Rdest, Rsrc1, Src2 | Adunare fara depasire |
| **addiu** Rdest, Rsrc1, Imm | Adunare imediata fara depasire |
| **SI logic** | |
| **and** Rdest, Rsrc1, Src2 | SI logic |
| **andi** Rdest, Rsrc1, Imm | SI logic cu o constanta imediata |
| **SAU logic** | |
| **or** Rdest, Rsrc1, Src2 | SAU logic |
| **ori** Rdest, Rsrc1, Imm | SAU logic cu o constanta imediata |
| **NOT** | |
| **not** Rdest, Rsrc | NOT |
| **XOR** | |
| **xor** Rdest, Rsrc1, Src2 | XOR |
| **xori** Rdest, Rsrc1, Imm | XOR imediat |
| **Modul** | |
| **abs** Rdest, Rsrc | Valoare absoluta |
| **Impartire**  *- Catul este pus in registrul LO si restul in registrul HI -* | |
| **div** Rsrc1, Rsrc2 | Impartire cu depasire |
| **divu** Rsrc1, Ssrc2 | Impartire fara depasire |
| **Inmultire**  *- Partea mai putin semnificativa a rezultatului in registrul LO, cea mai semnificativa in registrul HI -* | |
| **mult** Rsrc1, Src2 | Inmultire |
| **multu** Rsrc1, Src2 | Inmultire fara semn |
| **Scadere** | |
| **sub** Rdest, Rsrc1, Src2 | Scadere cu depasire |
| **subu** Rdest, Rsrc1, Src2 | Scadere fara depasire |
| **Rest** | |
| *rem* Rdest, Rsrc1, Src2 | Rest |
| *remu* Rdest, Rsrc1, Src2 | Rest fara semn |
| **Shift** | |
| *sll* Rdest, Rsrc1, Src2 | Shift logic la stanga variabil |
| *sra* Rdest, Rsrc1, Src2 | Shift aritmetic la dreapta |
| *srl* Rdest, Rsrc1, Src2 | Shift logic la dreapta variabil |
| **Rotire** | |
| *rol* Rdest, Rsrc1, Src2 | Rotire la stanga |
| *ror* Rdest, Rsrc1, Src2 | Rotire la dreapta |

## Instructiuni cu referire la memorie + manipulare constante

|  |  |
| --- | --- |
| **Op Dest, Src** | **Descriere** |
| **Load** | |
| **la** Rdest, address | Incarcare adresa |
| *lb* Rdest, address | Incarcare octet |
| *lbu* Rdest, address | Incarcare octet fara semn |
| *ld* Rdest, address | Incarcare dublu cuvant |
| *lh* Rdest, address | Incarcare semicuvant |
| *lhu* Rdest, address | Incarcare semicuvant fara semn |
| **lw** Rdest, address | Incarcare cuvant |
| *lwcz* Rdest, address | Incarcare cuvant in coprocesor |
| *lwl* Rdest, address | Incarcare din stanga a cuvantului |
| *lwr* Rdest, address | Incarcare din dreapta cuvantului |
| **li** Rdest, Imm | Incarcare imediata |
| *li.d* FRdest, float | Incarcare constanta imediata dubla precizie |
| *li.s* FRdest, float | Incarcare constanta imediata simpla precizie |
| **lui** Rdest, integer | Incarcare constanta intreaga |
| **Store** | |
| *sb* Rdest, address | Memorare octet |
| *sd* Rdest, address | Memorare dublu cuvant |
| *sh* Rdest, address | Memorare semicuvant |
| **sw** Rdest, address | Memorare cuvant |
| *swcz* Rdest, address | Memorare cuvant din coprocesor |
| *swl* Rdest, address | Memorare portiune stanga din cuvant |
| *swr* Rdest, address | Memorare portiune dreapta din cuvant |

## Instructiuni de intreruperi si exceptii

|  |  |
| --- | --- |
| **Op** | **Descriere** |
| *rfe* | Revenire din exceptie |
| **syscall** | Apel system( registrul $v0 contine numarul apelului ) |
| *break n* | Exceptia n |
| *nop* | No operation |

## Instructiuni de salt si ramificatie

|  |  |
| --- | --- |
| **Op Dest, Src1, (Src2)** | **Descriere** |
| *b* label | Salt neconditional la *label* |
| *bczt* label | Branch pe conditie adevarata in coprocessor |
| *bczf* label | Branch pe conditie falsa in coprocessor |
| *beq* Rsrc1, Src2, label | Branch pe egalitate |
| *bge* Rsrc1, Src2, label | Branch pe conditie de mai mare sau egal |
| *bgt* Rsrc1, Src2, label | Branch pe conditie de mai mare strict |
| *ble* Rsrc1, Src2, label | Branch pe conditie de mai mic sau egal |
| *blt* Rsrc1, Src2, label | Branch pe conditie de mai mic strict |
| *bne* Rsrc1, Src2, label | Branch pe neegalitate |
| **beqz** Rsrc, label | Branch pe egalitate cu 0 |
| *bgez* Rsrc, label | Branch pe conditie de mai mare sau egal cu 0 |
| *bgezal* Rsrc, label | Branch pe conditie de >= 0 si salvare legatura |
| *bltz* Rsrc, label | Branch pe conditie de < 0 |
| *bnez* Rsrc, label | Branch pe neegalitate de 0 |
| *bltzal* Rsrc, label | Branch pe conditie de < 0 si salvare legatura |
| *bgtz* Rsrc, label | Branch pe conditie de > 0 |
| **blez** Rsrc, label | Branch pe conditie de <= 0 |
| **j** label | Jump |
| **jal** label | Jump si salvare legatura |
| **jalr** Rsrc | Jump si salvare legatura prin registru |
| **jr** Rsrc | Jump la adresa din registru |

## Instructiuni de comparatie

|  |  |
| --- | --- |
| **Op Dest, Src1, (Src2)** | **Descriere** |
| *seq* Rdest, Rsrc1, Rsrc2 | Seteaza flag in caz de egalitate |
| *sge* Rdest, Rsrc1, Rsrc2 | Seteaza flag pe relatia de > sau = |
| *sgt* Rdest, Rsrc1, Rsrc2 | Seteaza flag pe relatia de > |
| *sle* Rdest, Rsrc1, Rsrc2 | Seteaza flag pe relatia de < sau = |
| *slt* Rdest, Rsrc1, Rsrc2 | Seteaza flag pe relatia de < |
| *slti* Rdest, Rsrc1, Imm | Seteaza flag pe relatia de < decat val imediata |
| *sne* Rdest, Rsrc1, Rsrc2 | Seteaza flag in caz de inegalitate |

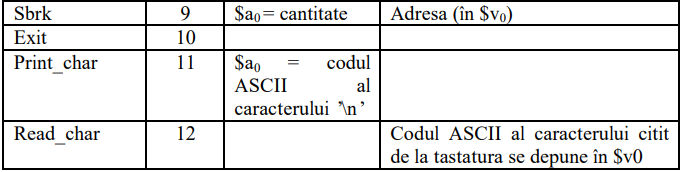
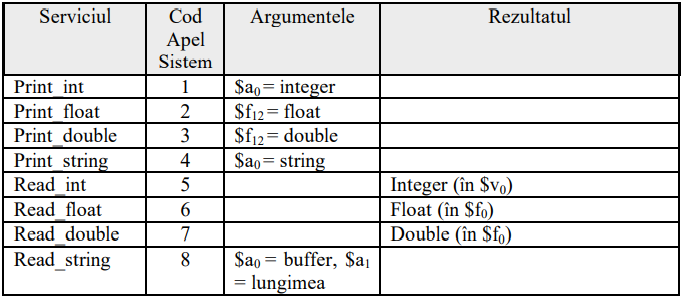
## Instructiuni de transfer

|  |  |
| --- | --- |
| **Op Dest, Src1, (Src2)** | **Descriere** |
| **move** Rdest, Rsrc | Transfer |
| **mfhi** Rdest | Transfera din HI |
| **mflo** Rdest | Transfera din LO |
| **mthi** Rdest | Transfera in HI |
| **mtlo** Rdest | Transfera in LO |
| *mfcz* Rdest, Copsource | Transfera din coprocesor z |
| *mfc1* Rdest, FRsrc1 | Transfer dublu cuvant flotant din coprocesorul1 |
| *mtcz* Rdest, Copdest | Transfer in coprocesor z |

## Instructiuni in virgula mobila

|  |  |
| --- | --- |
| **Op Dest, Src1, (Src2)** | **Descriere** |
| *abs.d* FRdest, FRsource | Valoare absoluta registru flotant simpla precizie |
| *abs.s* FRdest, FRsource | Valoare absoluta registru flotant dubla precizie |
| *add.d* FRdest, FRsrc1, FRsrc2 | Adunare in dubla precizie |
| *add.s* FRdest, FRsrc1, FRsrc2 | Adunare in simpla precizie |
| *c.le.d* FRdest, FRsrc1, FRsrc2 | Comparatie pe relatie de < sau = in dubla precizie |
| *c.le.s* FRdest, FRsrc1, FRsrc2 | Comparatie pe relatie de < sau = in simpla precizie |
| *cvt.d.s* FRdest, FRsource | Conversie din simpla in dubla precizie |
| *cvt.d.w* FRdest, FRsource | Conversie din intreg in dubla precizie |
| *cvt.s.d* FRdest, FRsource | Conversie din dubla in simpla precizie |
| *cvt.s.w* FRdest, FRsource | Conversie din intreg in simpla precizie |
| *cvt.w.d* FRdest, FRsource | Conversie din dubla precizie in intreg |
| *cvt.w.s* FRdest, FRsource | Conversie din simpla precizie in intreg |
| *div.d* FRdest, FRsrc1, FRsrc2 | Impartire in dubla precizie |
| *div.s* FRdest, FRsrc1, FRsrc2 | Impartire in simpla precizie |
| *l.d* FRdest, address | Incarcare valoare dubla precizie de la adresa |
| *l.s* FRdest, address | Incarcare valoare simpla precizie de la adresa |
| *mov.d* FRdest, FRsource | Transfera valoare dubla precizie |
| *mov.s* FRdest, FRsource | Transfera valoare simpla precizie |
| *mul.d* FRdest, FRsrc1, FRsrc2 | Inmultire in dubla precizie |
| *mul.s* FRdest, FRsrc1, FRsrc2 | Inmultire in simpla precizie |
| *s.d* FRdest, address | Memorare valoare dubla precizie la adresa |
| *s.s* FRdest, address | Memorare valoare simpla precizie la adresa |
| *sub.d* FRdest, FRsrc1, FRsrc2 | Scadere in dubla precizie |
| *sub.s* FRdest, FRsrc1, FRsrc2 | Scadere in simpla precizie |

# Apeluri sistem



**Sbrk -**  intoarce un pointer la un bloc de memorie ce contine n octeti

# Pseudoinstructiuni MIPS

Afisare string pe consola:

* **puts** ***label*** ( *label* – eticheta de la care se va afisa sirul de caractere)

Afisare carecter pe consola:

* **putc *vchar*** (*vchar* – caracterul care se va afisa pe consola)

Afisare numar intreg pe consola:

* **puti $a0** (argumentul este nr registrului al carui continut se va afisa pe consola)

Citire caracter de la tastatura:

* **getc** **$a0** (argumentul este registrul in care se va incarca codul ASCII al caracterului citit)

Citire intreg de la tastatura:

* **geti $a0** (argumentul este registrul in care se va incarca valoarea intreaga citita)

Terminare program:

* **done**

Apel sistem:

* **syscall**