MIPS32® Instruction Set Quick Reference

R

D

 D

ESTINATION REGISTER R

S

, R

T

 S

OURCE OPERAND REGISTERS R

A

 R

ETURN ADDRESS REGISTER

(R31) PC  P

ROGRAM COUNTER A

CC

 64-

BIT ACCUMULATOR L

O

, H

I

 A

CCUMULATOR LOW

(A

CC

31:0

)

AND HIGH

(A

CC

63:32

)

PARTS ±  S

IGNED OPERAND OR SIGN EXTENSION ∅  U

NSIGNED OPERAND OR ZERO EXTENSION ::  C

ONCATENATION OF BIT FIELDS R2  MIPS32 R

ELEASE

***L***

***OGICAL AND***

***B***

***IT***

***-F***

***IELD***

***O***

***PERATIONS***

***J***

***UMPS***

***A***

***ND***

***B***

***RANCHES***

***(N***

***OTE***

***: O***

***NE***

***D***

***ELAY***

***S***

***LOT***

***)***

AND R

D

, R

S

, R

T

R

D

= R

S

& R

T

B

OFF

18 PC +=

OFF

18±

ANDI R

D

, R

S

,

CONST

16 R

D

= R

S

&

CONST

16∅

BAL

OFF

EXTR2 R

D

18 R

A

= PC + 8, PC +=

OFF

18±

, R

S

, P, S R

S

= R

S

P+S-1:P

∅

BEQ R

S

, R

T

,

OFF

18

IF

R

S

= R

T

, PC +=

OFF

18±

INSR2 R

D

, R

S

, P, S R

D

P+S-1:P

= R

S

S-1:0

BEQZ R

S

,

OFF

18

IF

R

S

= 0, PC +=

OFF

18± NOP N

O

-

OP

BGEZ R

S NOR R

D

, R

S

, R

T

R

D

= ~(R

S

| R

T

)

NOT R

D

, R

S

R

D

= ~R

S

OR R

D

, R

S

, R

T

R

D

= R

S

| R

T

ORI R

D

, R

S

,

CONST

16 R

D

= R

S

|

CONST

16∅

WSBHR2 R

D

, R

S

R

D

= R

S

23:16

:: R

S

31:24

:: R

S

7:0

:: R

S

15:8

XOR R

D

,

OFF

18

IF

R

S

≥ 0, PC +=

OFF

18±

BGEZAL R

S

,

OFF

18 R

A

= PC + 8;

IF

R

S

≥ 0, PC +=

OFF

18±

BGTZ R

S

,

OFF

18

IF

R

S

> 0, PC +=

OFF

18±

2

INSTRUCTION

BLEZ R

S

,

OFF

18

IF

R

S

≤ 0, PC +=

OFF

18±

DOTTED

 A

SSEMBLER PSEUDO

-

INSTRUCTION

BLTZ R

S

,

OFF

18

IF

R

S

< 0, PC +=

OFF

18± P

LEASE REFER TO

*II: T*

*HE*

*“MIPS32 A*

*RCHITECTURE*

*F*

*OR*

*P*

*ROGRAMMERS*

*V*

*OLUME MIPS32 I*

*NSTRUCTION*

*S*

*ET*

*”*

FOR COMPLETE INSTRUCTION SET INFORMATION

.

, R

S

, R

T

R

D

= R

S

⊕ R

T

BLTZAL R

S

,

OFF

18 R

A

= PC + 8;

IF

R

S

< 0, PC +=

OFF

18±

BNE R

S

, R

T

,

OFF

18

IF

R

S

≠ R

T

, PC +=

OFF

18±

***A***

***RITHMETIC***

***O***

***PERATIONS***

XORI R

D

, R

S

,

CONST

16 R

D

= R

S

⊕

CONST

16∅

BNEZ R

S

,

OFF

18

IF

R

S

≠ 0, PC +=

OFF

18±

ADD R

D

, R

S

, R

T

R

D

= R

S

+ R

T

(

OVERFLOW TRAP

)

***C***

***ONDITION***

***O***

***PERATIONS***

J

ADDR

28 PC = PC

31:28

::

ADDR

28∅

ADDI R

D

MOVN R

D

JAL

ADDR T

***ESTING AND***

***C***

***ONDITIONAL***

***M***

***OVE , R***

S

,

CONST

16 R

D

= R

S

+

CONST

16± (

OVERFLOW TRAP

)

ADDIU R

D

, R

S

, R

T IF

R

T

≠ 0, R

D

= R

S

28 R

A

= PC + 8; PC = PC

31:28

::

ADDR

28∅

, R

S

,

CONST

16 R

D

= R

S

+

CONST

16±

MOVZ R

D

, R

S

, R

T IF

R

T

= 0, R

D

= R

S

JALR R

D

, R

S

R

D

= PC + 8; PC = R

S

ADDU R

D

, R

S

, R

T

R

D

= R

S

+ R

T

SLT R

D

, R

S

, R

T

R

D

= (R

S

± < R

T

±) ? 1 : 0

JR R

S

PC = R

S

CLO R

D

, R

S

R

D

= C

OUNT

L

EADING

O

NES

(R

S

)

CLZ R

D

, R

S

R

D

= C

OUNT

L

EADING

Z

EROS

(R

S

)

LA R

D

,

LABEL

R

D

= A

DDRESS

(

LABEL

)

SLTI R

D

, R

S

,

CONST

16 R

D

= (R

S

± <

CONST

16±) ? 1 : 0

***L***

***OAD AND***

***S***

***TORE***

***O***

***PERATIONS SLTIU R***

D

, R

S

,

CONST

16 R

D

= (R

S

∅ <

CONST

16∅) ? 1 : 0

LB R

D

,

OFF

16(R

S

) R

D

=

MEM

8(R

S

+

OFF

16±)±

LI R

D

,

IMM

32 R

D

=

IMM

32

SLTU R

D

, R

S

, R

T

R

D

= (R

S

∅ < R

T

∅) ? 1 : 0

LBU R

D

,

OFF

16(R

S

) R

D

=

MEM

8(R

S

+

OFF

16±)∅

LUI R

D

,

CONST

16 R

D

=

CONST

16 << 16

***M***

***ULTIPLY AND***

***O***

***PERATIONS***

LH R

D

,

MOVE R

D

OFF

16(R

S

) R

D

=

MEM

16(R

S

+

OFF

16±)±

DIV R

S

***D***

***IVIDE , R***

S

R

D

= R

S

NEGU R

D

, R

S

R

D

= –R

S

, R

T

L

O

= R

S

± / R

T

±; Η

Ι

= R

S

±

MOD

R

T

±

LHU R

D

,

OFF

16(R

S

) R

D

=

MEM

16(R

S

+

OFF

16±)∅

LW R

D SEBR2 R

D

, R

S

R

D

= R

S

7:0

±

SEHR2 R

D

, R

S

R

D

= R

S

15:0

±

SUB R

D

, R

S

, R

T

R

D

= R

S

– R

T

(

OVERFLOW TRAP

)

DIVU R

S

, R

T

L

O

= R

S

∅ / R

T

∅; Η

Ι

= R

S

∅

MOD

R

T

∅

MADD R

S

, R

T

A

CC

+= R

S

± × R

T

±

MADDU R

S

, R

T

A

CC

+= R

S

∅ × R

T

∅

,

OFF

16(R

S

) R

D

=

MEM

32(R

S

+

OFF

16±)

LWL R

D

,

OFF

16(R

S

) R

D

= L

OAD

W

ORD

L

EFT

(R

S

+

OFF

16±)

LWR R

D

,

OFF

16(R

S

) R

D

= L

OAD

W

ORD

R

IGHT

(R

S

+

OFF

16±)

SB R

S SUBU R

D

, R

S

, R

T

R

D

= R

S

– R

T

MSUB R

S

, R

T

A

CC

−= R

S

± × R

T

±

MSUBU R

S

, R

T

A

CC

−= R

S

∅ × R

T

∅

,

OFF

16(R

T

)

MEM

8(R

T

+

OFF

16±) = R

S

7:0

SH R

S

,

OFF

16(R

T

)

MEM

16(R

T

+

OFF

16±) = R

S

15:0

***S***

***HIFT AND***

***R***

***OTATE***

***O***

***PERATIONS***

MUL R

D

, R

S

, R

T

R

D

= R

S

± × R

T

±

SW R

S

,

ROTRR2 R

D

, R

S

,

BITS

5 R

D

= R

S

BITS5–1:0

:: R

S

31:BITS5

MULT R

S

, R

T

A

CC

= R

S

± × R

T

±

ROTRVR2 R

D

, R

S

, R

T

R

D

= R

S

RT4:0–1:0

:: R

S

31:RT4:0

MULTU R

S

, R

T

A

CC

= R

S

∅ × R

T

∅

OFF

16(R

T

)

MEM

32(R

T

+

OFF

16±) = R

S

SWL R

S

,

OFF

16(R

T

) S

TORE

W

ORD

L

EFT

(R

T

+

OFF

16±, R

S

)

SWR R

S

,

OFF

16(R

T

) S

TORE

W

ORD

R

IGHT

(R

T

+

OFF

16±, R

S

)

SLL R

D

, R

S

,

SHIFT

5 R

D

= R

S

<<

SHIFT

5

ULW R

D

,

OFF

16(R

S

) R

D

=

UNALIGNED

\_

MEM

32(R

S

+

OFF

16 ±)

SLLV R

D

, R

S

, R

T

R

D

= R

S

<< R

T

4:0

***A***

***CCUMULATOR***

***A***

***CCESS***

***O***

***PERATIONS***

USW R

S

,

OFF

16(R

T

)

UNALIGNED

\_

MEM

32(R

T

+

OFF

16±) = R

S

SRA R

D

, R

S

,

SHIFT

5 R

D

= R

S

± >>

SHIFT

5

MFHI R

D

R

D

= H

I

SRAV R

D

, R

S

, R

T

R

D

= R

S

± >> R

T

4:0

MFLO R

D

= L

O

***A***

***TOMIC***

***R***

MTHI R

S

R

D

***EAD***

***-M***

***ODIFY***

***-W***

***RITE***

***O***

***PERATIONS***

SRL R

D

, R

S

,

SHIFT

5 R

D

= R

S

∅ >>

SHIFT

5

H

I

= R

S

LL R

D

,

OFF

16(R

S

) R

D

=

MEM

32(R

S

+

OFF

16±);

LINK

SRLV R

D

, R

S

, R

T

R

D

= R

S

∅ >> R

T

4:0

MTLO R

S

L

O

= R

S

SC R

D

,

OFF

16(R

S

)

IF

A

TOMIC

,

MEM

32(R

S

+

OFF

16±) = R

D

; R

D

= A

TOMIC

? 1 : 0

Copyright © 2008 MIPS Technologies, Inc. All rights reserved. MD00565 Revision 01.01

***-W R***

***EGISTERS***

***R***

***EADING***

***T***

***HE***

***C***

***YCLE***

***C***

***OUNT***

***R***

***EGISTER***

***F***

***ROM***

***C***

***A***

***TOMIC***

***R***

***EAD***

***-M***

***ODIFY***

***RITE***

0 zero Always equal to zero

1 at Assembler temporary; used by the assembler

unsigned mips\_cycle\_counter\_read() { 2-3 v0-v1 Return value from a function call

unsigned cc; asm volatile("mfc0 %0, $9" : "=r" (cc)); 4-7 a0-a3 First four parameters for a function call

}

return (cc << 1);

8-15 t0-t7 Temporary variables; need not be preserved

16-23 s0-s7 Function variables; must be preserved

24-25 t8-t9 Two more temporary variables

***A***

***SSEMBLY***

26-27 k0-k1 Kernel use registers; may change unexpectedly

28 gp Global pointer

29 sp Stack pointer

30 fp/s8 Stack frame pointer or subroutine variable

31 ra Return address of the last subroutine call

***D***

***EFAULT***

***E***

***XAMPLE***

atomic\_inc:

ll $t0, 0($a0) # load linked addiu $t1, $t0, 1 # increment sc $t1, 0($a0) # store cond'l beqz $t1, atomic\_inc # loop if failed nop

***-L***

***ANGUAGE***

***F***

***UNCTION***

***E***

***XAMPLE***

***A***

***CCESSING***

***U***

***NALIGNED***

***D***

***ATA***

***NOTE***

***: ULW***

***AND***

***USW***

***AUTOMATICALLY GENERATE APPROPRIATE CODE***

# int asm\_max(int a, int b) # { # int r = (a < b) ? b : a;

***L***

***ITTLE***

***-E***

***NDIAN***

***M***

***ODE***

***B***

***IG***

***-E***

***NDIAN***

***M***

***ODE***

LWR R

D

,

OFF

16(R

S

) # return r;

LWL R

D # }

.text .set nomacro .set noreorder

.global asm\_max .ent asm\_max

asm\_max:

move $v0, $a0 # r = a slt $t0, $a0, $a1 # a < b ? jr $ra # return movn $v0, $a1, $t0 # if yes, r = b

.end asm\_max

***C / A***

***SSEMBLY***

LWL R

D

,

OFF

16(R

S

) ,

OFF

16+3(R

S

)

LWR R

D

,

OFF

16+3(R

S

)

SWR R

D

,

OFF

16(R

S

) SWL R

D

SWL R

D

,

OFF

16(R

S

) ,

OFF

16+3(R

S

)

SWR R

D

,

OFF

16+3(R

S

)

***C C***

***ALLING***

***C***

***ONVENTION***

***(O32)***

***A***

***CCESSING***

***U***

***NALIGNED***

***D***

***ATA***

***F***

***ROM***

***C***

**Stack Management**

• The stack grows down.

• • Subtract from $sp to allocate local storage space.

Restore $sp by adding the same amount at function exit.

typedef struct {

int u; } \_\_attribute\_\_((packed)) unaligned;

• The stack must be 8-byte aligned.

• Modify $sp only in multiples of eight.

int unaligned\_load(void \*ptr) { Function Parameters

• Every parameter smaller than 32 bits is promoted to 32 bits.

• First four parameters are passed in registers $a0−$a3.

unaligned \*uptr = (unaligned \*)ptr; return uptr->u; }

• 64-bit parameters are passed in register pairs:

• Little-endian mode: $a1:$a0 or $a3:$a2.

• Big-endian mode: $a0:$a1 or $a2:$a3.

• Every subsequent parameter is passed through the stack.

***MIPS SDE-GCC C***

***OMPILER***

• • First 16 bytes on the stack are not used.

Assuming $sp was not modified at function entry:

• The 1st stack parameter is located at 16($sp).

• The 2nd stack parameter is located at 20($sp), etc.

• 64-bit parameters are 8-byte aligned.

**Return Values**

• 32-bit and smaller values are returned in register $v0.

• 64-bit values are returned in registers $v0 and $v1:

• Little-endian mode: $v1:$v0.

• Big-endian mode: $v0:$v1.

***MIPS32 V***

***IRTUAL***

***I***

***NTERFACE***

#include <stdio.h>

int asm\_max(int a, int b);

int main() {

int x = asm\_max(10, 100); int y = asm\_max(200, 20); printf("%d %d\n", x, y); }

***I***

***NVOKING***

***-L***

***ANGUAGE***

***F***

***UNCTION***

***D***

***EFINES***

\_\_mips MIPS ISA (= 32 for MIPS32)

\_\_mips\_isa\_rev MIPS ISA Revision (= 2 for MIPS32 R2)

\_\_mips\_dsp DSP ASE extensions enabled

\_MIPSEB Big-endian target CPU

\_MIPSEL Little-endian target CPU

\_MIPS\_ARCH\_CPU Target CPU specified by -march=CPU

\_MIPS\_TUNE\_CPU Pipeline tuning selected by -mtune=CPU C

int dp(int a[], int b[], int n)

***N***

***OTES***

{

int i; long long acc = (long long) a[0] \* b[0]; for (i = 1; i < n; i++)

acc += (long long) a[i] \* b[i]; return (acc >> 31); }

***MULT A***

***ND***

***MADD I***

***NSTRUCTIONS***

***F***

***ROM***

***A***

***DDRESS***

***S***

***PACE***

• Many assembler pseudo-instructions and some rarely used kseg3 0xE000.0000 0xFFFF.FFFF Mapped Cached

machine instructions are omitted.

ksseg 0xC000.0000 0xDFFF.FFFF Mapped Cached

• The C calling convention is simplified. Additional rules apply when passing complex data structures as function parameters. kseg1 0xA000.0000 0xBFFF.FFFF Unmapped Uncached

kseg0 0x8000.0000 0x9FFF.FFFF Unmapped Cached

• The examples illustrate syntax used by GCC compilers.

• Most MIPS processors increment the cycle counter every other cycle. Please check your processor documentation. useg 0x0000.0000 0x7FFF.FFFF Mapped Cached

Copyright © 2008 MIPS Technologies, Inc. All rights reserved.

MD00565 Revision 01.01