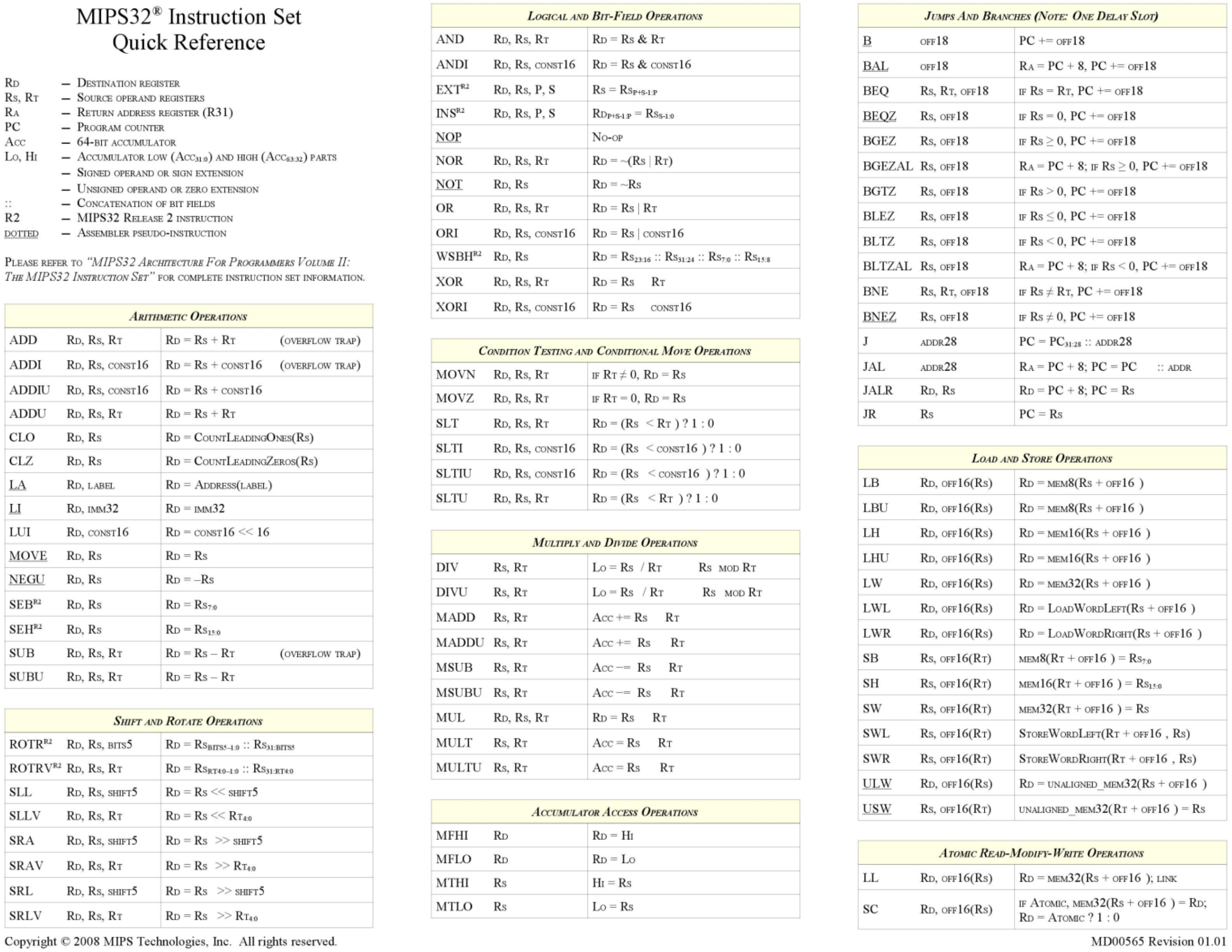
ECE/CS 250 EXAM REFERENCE SHEET

(You may keep or discard this sheet)

MIPS info



MIPS32® Instruction Set Quick Reference

Rd — Destination register
Rs, Rt — Source operand registers
Ra — Return address register (R31)
PC — Program counter
Acc — 64-bit accumulator
Lo, Hi — Accumulator low ($Acc_{31:0}$) and high ($Acc_{63:32}$) parts
— Signed operand or sign extension
— Unsigned operand or zero extension
:: — Concatenation of bit fields
R2 — MIPS32 Release 2 instruction
dotted — Assembler pseudo-instruction

Please refer to *"MIPS32 Architecture For Programmers Volume II: The MIPS32 Instruction Set"* for complete instruction set information.

Arithmetic Operations

ADD	Rd, Rs, Rt	Rd = Rs + Rt (overflow trap)
ADDI	Rd, Rs, const16	Rd = Rs + const16 (overflow trap)
ADDIU	Rd, Rs, const16	Rd = Rs + const16
ADDU	Rd, Rs, Rt	Rd = Rs + Rt
CLO	Rd, Rs	Rd = CountLeadingOnes(Rs)
CLZ	Rd, Rs	Rd = CountLeadingZeros(Rs)
LA	Rd, label	Rd = Address(label)
LI	Rd, imm32	Rd = imm32
LUI	Rd, const16	Rd = const16 << 16
MOVE	Rd, Rs	Rd = Rs
NEGU	Rd, Rs	Rd = –Rs
SEB[R2]	Rd, Rs	Rd = $Rs_{7:0}$
SEH[R2]	Rd, Rs	Rd = $Rs_{15:0}$
SUB	Rd, Rs, Rt	Rd = Rs – Rt (overflow trap)
SUBU	Rd, Rs, Rt	Rd = Rs – Rt

Shift and Rotate Operations

ROTR[R2]	Rd, Rs, bits5	Rd = $Rs_{bits5-1:0}$ :: $Rs_{31:bits5}$
ROTRV[R2]	Rd, Rs, Rt	Rd = $Rs_{Rt4:0-1:0}$ :: $Rs_{31:Rt4:0}$
SLL	Rd, Rs, shift5	Rd = Rs << shift5
SLLV	Rd, Rs, Rt	Rd = Rs << $Rt_{4:0}$
SRA	Rd, Rs, shift5	Rd = Rs >> shift5
SRAV	Rd, Rs, Rt	Rd = Rs >> $Rt_{4:0}$
SRL	Rd, Rs, shift5	Rd = Rs >> shift5
SRLV	Rd, Rs, Rt	Rd = Rs >> $Rt_{4:0}$

Logical and Bit-Field Operations

AND	Rd, Rs, Rt	Rd = Rs & Rt
ANDI	Rd, Rs, const16	Rd = Rs & const16
EXT[R2]	Rd, Rs, P, S	Rs = $Rs_{P+S-1:P}$
INS[R2]	Rd, Rs, P, S	$Rd_{P+S-1:P}$ = $Rs_{S-1:0}$
NOP		No-op
NOR	Rd, Rs, Rt	Rd = ~(Rs | Rt)
NOT	Rd, Rs	Rd = ~Rs
OR	Rd, Rs, Rt	Rd = Rs | Rt
ORI	Rd, Rs, const16	Rd = Rs | const16
WSBH[R2]	Rd, Rs	Rd = $Rs_{23:16}$ :: $Rs_{31:24}$ :: $Rs_{7:0}$ :: $Rs_{15:8}$
XOR	Rd, Rs, Rt	Rd = Rs Rt
XORI	Rd, Rs, const16	Rd = Rs const16

Condition Testing and Conditional Move Operations

MOVN	Rd, Rs, Rt	if Rt ≠ 0, Rd = Rs
MOVZ	Rd, Rs, Rt	if Rt = 0, Rd = Rs
SLT	Rd, Rs, Rt	Rd = (Rs < Rt) ? 1 : 0
SLTI	Rd, Rs, const16	Rd = (Rs < const16) ? 1 : 0
SLTIU	Rd, Rs, const16	Rd = (Rs < const16) ? 1 : 0
SLTU	Rd, Rs, Rt	Rd = (Rs < Rt) ? 1 : 0

Multiply and Divide Operations

DIV	Rs, Rt	Lo = Rs / Rt Rs mod Rt
DIVU	Rs, Rt	Lo = Rs / Rt Rs mod Rt
MADD	Rs, Rt	Acc += Rs Rt
MADDU	Rs, Rt	Acc += Rs Rt
MSUB	Rs, Rt	Acc −= Rs Rt
MSUBU	Rs, Rt	Acc −= Rs Rt
MUL	Rd, Rs, Rt	Rd = Rs Rt
MULT	Rs, Rt	Acc = Rs Rt
MULTU	Rs, Rt	Acc = Rs Rt

Accumulator Access Operations

MFHI	Rd	Rd = Hi
MFLO	Rd	Rd = Lo
MTHI	Rs	Hi = Rs
MTLO	Rs	Lo = Rs

Jumps And Branches (Note: One Delay Slot)

B	off18	PC += off18
BAL	off18	Ra = PC + 8, PC += off18
BEQ	Rs, Rt, off18	if Rs = Rt, PC += off18
BEQZ	Rs, off18	if Rs = 0, PC += off18
BGEZ	Rs, off18	if Rs ≥ 0, PC += off18
BGEZAL	Rs, off18	Ra = PC + 8; if Rs ≥ 0, PC += off18
BGTZ	Rs, off18	if Rs > 0, PC += off18
BLEZ	Rs, off18	if Rs ≤ 0, PC += off18
BLTZ	Rs, off18	if Rs < 0, PC += off18
BLTZAL	Rs, off18	Ra = PC + 8; if Rs < 0, PC += off18
BNE	Rs, Rt, off18	if Rs ≠ Rt, PC += off18
BNEZ	Rs, off18	if Rs ≠ 0, PC += off18
J	addr28	PC = $PC_{31:28}$ :: addr28
JAL	addr28	Ra = PC + 8; PC = PC :: addr
JALR	Rd, Rs	Rd = PC + 8; PC = Rs
JR	Rs	PC = Rs

Load and Store Operations

LB	Rd, off16(Rs)	Rd = mem8(Rs + off16)
LBU	Rd, off16(Rs)	Rd = mem8(Rs + off16)
LH	Rd, off16(Rs)	Rd = mem16(Rs + off16)
LHU	Rd, off16(Rs)	Rd = mem16(Rs + off16)
LW	Rd, off16(Rs)	Rd = mem32(Rs + off16)
LWL	Rd, off16(Rs)	Rd = LoadWordLeft(Rs + off16)
LWR	Rd, off16(Rs)	Rd = LoadWordRight(Rs + off16)
SB	Rs, off16(Rt)	mem8(Rt + off16) = $Rs_{7:0}$
SH	Rs, off16(Rt)	mem16(Rt + off16) = $Rs_{15:0}$
SW	Rs, off16(Rt)	mem32(Rt + off16) = Rs
SWL	Rs, off16(Rt)	StoreWordLeft(Rt + off16 , Rs)
SWR	Rs, off16(Rt)	StoreWordRight(Rt + off16 , Rs)
ULW	Rd, off16(Rs)	Rd = unaligned_mem32(Rs + off16)
USW	Rs, off16(Rt)	unaligned_mem32(Rt + off16) = Rs

Atomic Read-Modify-Write Operations

LL	Rd, off16(Rs)	Rd = mem32(Rs + off16); link

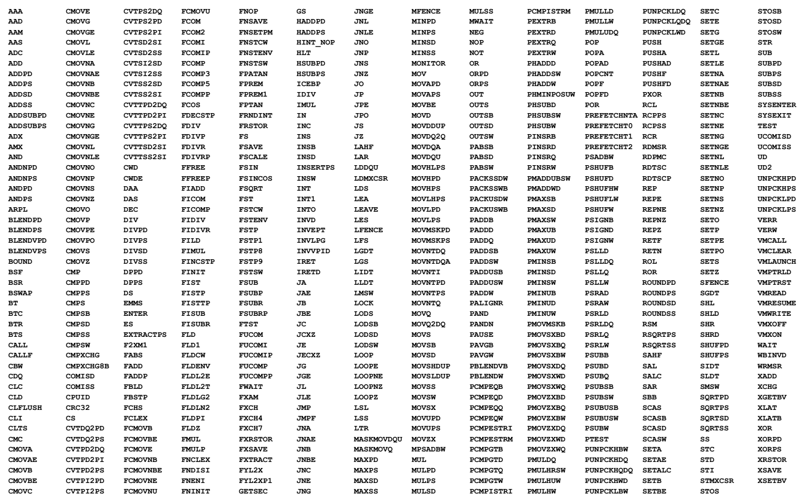
SC	Rd, off16(Rs)	if Atomic, mem32(Rs + off16) = Rd;
		Rd = Atomic ? 1 : 0

MD00565 Revision 01.01

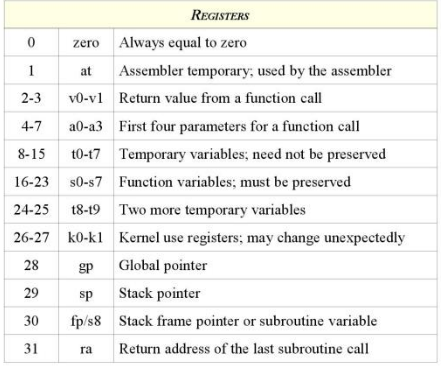
Powers of two

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1,024
11	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536

The Intel x86 instruction set



AAA
AAD
AAM
AAS
ADC
ADD
ADDPD
ADDPS
ADDSD
ADDSS
ADDSUBPD
ADDSUBPS
ADX
AMX
AND
ANDNPD
ANDNPS
ANDPD
ANDPS
ARPL
BLENDPD
BLENDPS
BLENDVPD
BLENDVPS
BOUND
BSF
BSR
BSWAP
BT
BTC
BTR
BTS
CALL
CALLF
CBW
CDQ
CLC
CLD
CLFLUSH
CLI
CLTS
CMC
CMOVA
CMOVAE
CMOVB
CMOVBE
CMOVC
CMOVE
CMOVG
CMOVGE
CMOVL
CMOVLE
CMOVNA
CMOVNAE
CMOVNB
CMOVNBE
CMOVNC
CMOVNE
CMOVNG
CMOVNGE
CMOVNL
CMOVNLE
CMOVNO
CMOVNP
CMOVNS
CMOVNZ
CMOVO
CMOVP
CMOVPE
CMOVPO
CMOVS
CMOVZ
CMP
CMPPD
CMPPS
CMPS
CMPSB
CMPSD
CMPSS
CMPSW
CMPXCHG
CMPXCHG8B
COMISD
COMISS
CPUID
CRC32
CS
CVTDQ2PD
CVTDQ2PS
CVTPD2DQ
CVTPD2PI
CVTPD2PS
CVTPI2PD
CVTPI2PS
CVTPS2DQ
CVTPS2PD
CVTPS2PI
CVTSD2SI
CVTSD2SS
CVTSI2SD
CVTSI2SS
CVTSS2SD
CVTSS2SI
CVTTPD2DQ
CVTTPD2PI
CVTTPS2DQ
CVTTPS2PI
CVTTSD2SI
CVTTSS2SI
CWD
CWDE
DAA
DAS
DEC
DIV
DIVPD
DIVPS
DIVSD
DIVSS
DPPD
DPPS
DS
EMMS
ENTER
ES
EXTRACTPS
F2XM1
FABS
FADD
FADDP
FBLD
FBSTP
FCHS
FCLEX
FCMOVB
FCMOVBE
FCMOVE
FCMOVNB
FCMOVNBE
FCMOVNE
FCMOVNU
FCMOVU
FCOM
FCOM2
FCOMI
FCOMIP
FCOMP
FCOMP3
FCOMP5
FCOMPP
FCOS
FDECSTP
FDIV
FDIVP
FDIVR
FDIVRP
FFREE
FFREEP
FIADD
FICOM
FICOMP
FIDIV
FIDIVR
FILD
FIMUL
FINCSTP
FINIT
FIST
FISTP
FISTTP
FISUB
FISUBR
FLD
FLD1
FLDCW
FLDENV
FLDL2E
FLDL2T
FLDLG2
FLDLN2
FLDPI
FLDZ
FMUL
FMULP
FNCLEX
FNDISI
FNENI
FNINIT
FNOP
FNSAVE
FNSETPM
FNSTCW
FNSTENV
FNSTSW
FPATAN
FPREM
FPREM1
FPTAN
FRNDINT
FRSTOR
FS
FSAVE
FSCALE
FSIN
FSINCOS
FSQRT
FST
FSTCW
FSTENV
FSTP
FSTP1
FSTP8
FSTP9
FSTSW
FSUB
FSUBP
FSUBR
FSUBRP
FTST
FUCOM
FUCOMI
FUCOMIP
FUCOMP
FUCOMPP
FWAIT
FXAM
FXCH
FXCH4
FXCH7
FXRSTOR
FXSAVE
FXTRACT
FYL2X
FYL2XP1
GETSEC
GS
HADDPD
HADDPS
HINT_NOP
HLT
HSUBPD
HSUBPS
ICEBP
IDIV
IMUL
IN
INC
INS
INSB
INSD
INSERTPS
INSW
INT
INT1
INTO
INVD
INVEPT
INVLPG
INVVPID
IRET
IRETD
JA
JAE
JB
JBE
JC
JCXZ
JE
JECXZ
JG
JGE
JL
JLE
JMP
JMPF
JNA
JNAE
JNB
JNBE
JNC
JNE
JNG
JNGE
JNL
JNLE
JNO
JNP
JNS
JNZ
JO
JP
JPE
JPO
JS
JZ
LAHF
LAR
LDDQU
LDMXCSR
LDS
LEA
LEAVE
LES
LFENCE
LFS
LGDT
LGS
LIDT
LLDT
LMSW
LOCK
LODS
LODSB
LODSD
LODSW
LOOP
LOOPE
LOOPNE
LOOPNZ
LOOPZ
LSL
LSS
LTR
MASKMOVDQU
MASKMOVQ
MAXPD
MAXPS
MAXSD
MAXSS
MFENCE
MINPD
MINPS
MINSD
MINSS
MONITOR
MOV
MOVAPD
MOVAPS
MOVBE
MOVD
MOVDDUP
MOVDQ2Q
MOVDQA
MOVDQU
MOVHLPS
MOVHPD
MOVHPS
MOVLHPS
MOVLPD
MOVLPS
MOVMSKPD
MOVMSKPS
MOVNTDQ
MOVNTDQA
MOVNTI
MOVNTPD
MOVNTPS
MOVNTQ
MOVQ
MOVQ2DQ
MOVS
MOVSB
MOVSD
MOVSHDUP
MOVSLDUP
MOVSS
MOVSW
MOVSX
MOVUPD
MOVUPS
MOVZX
MPSADBW
MUL
MULPD
MULPS
MULSD
MULSS
MWAIT
NEG
NOP
NOT
OR
ORPD
ORPS
OUT
OUTS
OUTSB
OUTSD
OUTSW
PABSB
PABSD
PABSW
PACKSSDW
PACKSSWB
PACKUSDW
PACKUSWB
PADDB
PADDD
PADDQ
PADDSB
PADDSW
PADDUSB
PADDUSW
PADDW
PALIGNR
PAND
PANDN
PAUSE
PAVGB
PAVGW
PBLENDVB
PBLENDW
PCMPEQB
PCMPEQD
PCMPEQQ
PCMPEQW
PCMPESTRI
PCMPESTRM
PCMPGTB
PCMPGTD
PCMPGTQ
PCMPGTW
PCMPISTRI
PCMPISTRM
PEXTRB
PEXTRD
PEXTRQ
PEXTRW
PHADDD
PHADDSW
PHADDW
PHMINPOSUW
PHSUBD
PHSUBSW
PHSUBW
PINSRB
PINSRD
PINSRQ
PINSRW
PMADDUBSW
PMADDWD
PMAXSB
PMAXSD
PMAXSW
PMAXUB
PMAXUD
PMAXUW
PMINSB
PMINSD
PMINSW
PMINUB
PMINUD
PMINUW
PMOVMSKB
PMOVSXBD
PMOVSXBQ
PMOVSXBW
PMOVSXDQ
PMOVSXWD
PMOVSXWQ
PMOVZXBD
PMOVZXBQ
PMOVZXBW
PMOVZXDQ
PMOVZXWD
PMOVZXWQ
PMULDQ
PMULHRSW
PMULHUW
PMULHW
PMULLD
PMULLW
PMULUDQ
POP
POPA
POPAD
POPCNT
POPF
POPFD
POR
PREFETCHNTA
PREFETCHT0
PREFETCHT1
PREFETCHT2
PSADBW
PSHUFB
PSHUFD
PSHUFHW
PSHUFLW
PSHUFW
PSIGNB
PSIGND
PSIGNW
PSLLD
PSLLDQ
PSLLQ
PSLLW
PSRAD
PSRAW
PSRLD
PSRLDQ
PSRLQ
PSRLW
PSUBB
PSUBD
PSUBQ
PSUBSB
PSUBSW
PSUBUSB
PSUBUSW
PSUBW
PTEST
PUNPCKHBW
PUNPCKHDQ
PUNPCKHQDQ
PUNPCKHWD
PUNPCKLBW
PUNPCKLDQ
PUNPCKLQDQ
PUNPCKLWD
PUSH
PUSHA
PUSHAD
PUSHF
PUSHFD
PXOR
RCL
RCPPS
RCPSS
RCR
RDMSR
RDPMC
RDTSC
RDTSCP
REP
REPE
REPNE
REPNZ
REPZ
RETF
RETN
ROL
ROR
ROUNDPD
ROUNDPS
ROUNDSD
ROUNDSS
RSM
RSQRTPS
RSQRTSS
SAHF
SAL
SALC
SAR
SBB
SCAS
SCASB
SCASD
SCASW
SETA
SETAE
SETALC
SETB
SETBE
SETC
SETE
SETG
SETGE
SETL
SETLE
SETNA
SETNAE
SETNB
SETNBE
SETNC
SETNE
SETNG
SETNGE
SETNL
SETNLE
SETNO
SETNP
SETNS
SETNZ
SETO
SETP
SETPE
SETPO
SETS
SETZ
SFENCE
SGDT
SHL
SHLD
SHR
SHRD
SHUFPD
SHUFPS
SIDT
SLDT
SMSW
SQRTPD
SQRTPS
SQRTSD
SQRTSS
SS
STC
STD
STI
STMXCSR
STOS
STOSB
STOSD
STOSW
STR
SUB
SUBPD
SUBPS
SUBSD
SUBSS
SYSENTER
SYSEXIT
TEST
UCOMISD
UCOMISS
UD
UD2
UNPCKHPD
UNPCKHPS
UNPCKLPD
UNPCKLPS
VERR
VERW
VMCALL
VMCLEAR
VMLAUNCH
VMPTRLD
VMPTRST
VMREAD
VMRESUME
VMWRITE
VMXOFF
VMXON
WAIT
WBINVD
WRMSR
XADD
XCHG
XGETBV
XLAT
XLATB
XOR
XORPD
XORPS
XRSTOR
XSAVE
XSETBV



Registers

0	zero	Always equal to zero
1	at	Assembler temporary; used by the assembler
2-3	v0-v1	Return value from a function call
4-7	a0-a3	First four parameters for a function call
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
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