Question: You are to design an instruction set architecture (ISA) for a ne...

(2 bookmarks)

You are to design an instruction set architecture (ISA) for a new 16-bit microprocessor (µP). Your ISA is to be designed using RISC design principles, with the primary design goals being low cost and a minimal number of clock cycles per instruction. Following are the requirements for your ISA.
1. Provide the following information about your ISA:
a. List and describe the roles of the user-programmable registers.
b. List and describe the different instruction formats used.
c. For each instruction in your instruction set, list the following:
d. Assembly language for each form of the instruction - mnemonic and operands
e. Machine language for each form of the instruction: instruction code format, op-code, and operand encoding
f. Justification for including each form of the instruction in your ISA
2. For each C construct listed in item 6 above, provide an example showing how the construct would be "compiled", i.e. implemented with your instruction set, by writing an example of the C construct and the corresponding assembly language implementation.

View comments (1)

Expert Answer

Anonymous answered this
21 answers

Was this answer helpful? 0 0

Answer:

Introduction:

MIPS is a reduced instructions set computer (RISC) architecture. It is one of the first RISC Instruction set architectures. MIPS is an acronym for "Microprocessor without interlocked pipeline stages". It was developed by a team led by John Hennessey at Stanford University. MIPSnimplementations are primarily used in embedded systems such as Windows CE devices, routers, residential gateways, and video game consoles such as the Sony PlayStation 2 and PlayStation Portable. Until late 2006, they were also used in many of SGI's computer products. MIPS implementations were also used by Digital Equipment Corporation, NEC,Pyramid Technology, Siemens Nixdorf, Tandem Computers and others during the late 1980s and 1990s. Since MIPS is a RISC computer it employs less number of transistors and hence decreases the transistor count. Pipelining is thus heavily employed to make use of extra available space on the chip to improve code execution performance. MIPS was defined to be a 32 bit architecture called MIPS32. Later Revisions of this architecture is 64 bit in size and hence called MIPS64.

MIPS-16 INSTRUCTION SET

Sr. No.	Mnemonic Instruction	Format	Description
1	ADD	ADD Rs1, Rs2 ,Rd	Adds Rs1 and Rs2 and stores the sum in Rd ignoring carry.
2	ADC	ADC Rs1, Rs2 ,Rd	Adds Rs1 and Rs2 and stores the sum in Rd with previous carry.
3	SUB	SUB Rs1, Rs2 ,Rd	Subtracts Rs2 from Rs1 and stores the difference in Rd ignoring the previous borrow
4	SBB	SUB Rs1, Rs2 ,Rd	Subtracts Rs2 from Rs1and stores the difference in Rd with the previous borrow.
5	AND	AND Rs1, Rs2 ,Rd	Performs Bitwise AND of Rs1 and Rs2 and stores the result in Rd
6	OR	OR Rs1, Rs2 ,Rd	Performs Bitwise OR of Rs1 and Rs2 and stores the result in Rd
7	XOR	XOR Rs1, Rs2 ,Rd	Performs Bitwise XOR of Rs1 and Rs2 and stores the result in Rd
8	NOT	NOT Rs1 ,Rd	Performs Complement of Rs1 and stores the result in Rd
9	SHIFTL	SHIFTL	Shifts Rs1 by one place to the left and store it

		Rs1 ,Rd	it in R
11	ADDI	ADDI Rs1 ,Rd,#5-bit	Adds a 5-bit unsigned value to Rs1 and stores the sum in R
12	SUBI	SUBI Rs1 ,Rd,#5-bit	Subtracts a 5-bit unsigned value from Rs1 and stores the difference in Rd
13	MOV	MOV Rs1 ,Rd	Copies Rs1 to Rd
14	MVIH	MVIH Rd,#8-bit	Copies immediate value into higher byte of Rd
15	MVIL	MVIL Rd ,#8-bit	Copies immediate value into lower byte of Rd
16	LDIDR	LDIDR Rs1 ,Rd,#5-bit	Loads Rd with a nibble at address given by [Rs1 +5 bit immediate value]
17	STIDR	STIDR Rs1 ,Rd,#5-bit	Stores Rd with a nibble at address given by [Rs1 +5 bit immediate value]
18	LDIDX	LDIDX Rs1, Rs2 ,Rd	Loads Rd with a nibble at address given by [Rs1 + Rs2]
19	STIDX	STIDX Rs1, Rs2 ,Rd	Stores Rd with a nibble at address given by [Rs1 + Rs2
20	JMP	JMP #11-bit	Unconditional jump to address offset by 11 bit signed value from current PC value
21	JMPI	JMPI Rd ,#15	Unconditional jump to address offset by 5 bit signed value added to R
22	JGEO	JGEO Rs1 ,Rs2,#5-bit	Conditional Jump to PC + 5 bit signed offset if Rs1 is greater than or equal to Rs2
23	JLEO	JLEO Rs1 ,Rs2,#5-bit	Conditional Jump to PC + 5 bit signed offset if Rs1 is less than or equal to Rs2
24	JCO	JCO #5-bit	Conditional Jump to PC + 5 bit signed offset if carry is set
25	JEO	JEO Rs1 ,Rs2,#5-bit	Conditional Jump to PC + 5 bit signed offset if Rs1 equals to Rs2
26	PUSH	PUSH Rs1	Push Rs1 to the stack top and update stack top
27	POP	POP Rd	Pop from the stack top and store the value to Rd and update stack top
28	CALL	CALL Rs1	Calls a subroutine located at [Rs1]. Return address is pushed onto stack
29	JAL	JAL #11-bit	Calls a subroutine located at [PC + 11 bit signed offset].Return address is pushed onto stack.
30	MOVSP	MOVSP Rs1	Copies value at Rs1 to stack pointer SP
31	RET	RET	Return from a function. Return address is popped from the stack
32	STC	STC	Set the carry flag
33	NOP	NOP	No operation. Idle machine cycle should be executed
34	HLT	HLT	Halts the processor.
35	RST	RST	Resets the processor
36	IE	IE	Enables the interrupt