R-type:

General instruction format:

|  |  |  |  |
| --- | --- | --- | --- |
| Op  5 | Addr.  16 | Extra Space  6 | Func.  5 |

Add: add Addr. add the value in an address of a memory to the value in register, and then save the calculated value in register. Opcode/funct: 00/00

And: and Addr. do “and” operation between the value in an address of a memory and the value in register, and then save the calculated value in register. Opcode/funct: 01/01

Or: or Addr. do “or” operation between the value in an address of a memory and the value in register, and then save the calculated value in register. Opcode/funct: 02/02

Sll:

Slr:

Sub: sub Addr. subtract the value in an address of a memory from the value in register, and then save the calculated value in register. Opcode/funct: 03/03

Slt: slt Addr. Compare the value in an address of a memory with the value in register, and if the value in register is smaller, set the value of register to 1, otherwise, set it to 0

Opcode/funct: 03/03

Jr: jr Addr. Unconditionally jump to the instruction whose address is in the register.

I-type:

|  |  |  |  |
| --- | --- | --- | --- |
| Op  5 | Imm  16 | Extra Space  6 | Func.  5 |

Addi: addi Imm. add the immediate value to the value in register, and then save the calculated value in register. Opcode/funct: 04/04

Andi: andi Imm. do “and” operation between immediate value and the value in register, and then save the calculated value in register. Opcode/funct: 05/05

Ori: ori Imm. do “or” operation between immediate value and the value in register, and then save the calculated value in register. Opcode/funct: 06/06

Slti: slti Imm. Compare the immediate value with the value in register, and if the value in register is smaller, set the value of register to 1, otherwise, set it to 0

Opcode/funct: 07/07

Lui: lui Imm.

Opcode/funct: 0C/0C

J-type:

|  |  |
| --- | --- |
| Op  5 | Addr.  27 |

Lw: lw Addr. Load the 16 bit word at an address of memory into the register.

Opcode/funct: 0A/0A

Sw: sw Addr. Store the 16 bit word of the register into at an address of memory the register. Opcode/funct: 0B/0B

Jr: jr Addr. Unconditionally jump to the instruction at the specified address.

Opcode/funct: 0D/0D

Jal: jal Addr. Unconditionally jump to the instruction at the specified address. Save the address of the next instruction in the register.

Opcode/funct: 0E/0E

B-type:

|  |  |  |
| --- | --- | --- |
| Op  5 | Addr.  16 | Target  11 |

Beq: beq Addr. DO Compare the value in an address of a memory with the value in register, if they are equal, branch. Opcode/funct: 08/08

Bne: bne Addr. DO Compare the value in an address of a memory with the value in register, if they are not equal, branch. Opcode/funct: 09/09

Euclid’s algorithm

int gcd(int a, int b){

if(a==0){

return b;

}

while(b!=0){

if(a>b){

a=a-b;

}else{

b=b-a;

}

}

return a;

}

lw a

beq 0 Exitb

Loop:

lw b

beq 0 Exita

slt a

beq 1 AMinus

beq 0 BMinus

AMinus:

lw a

sub b

sw a

j Loop

BMinus:

lw b

sub a

sw b

j Loop

Exita:

lw a

j return

Exitb:

lw b

j return

Assembly Code

C Code

Load/Store word&Exception

lw a

addi 2

slt 5

beq 1 Num

bne 1 Leave

Num:

lw a

addi 10

sw a

Leave:

addi 0x8fff

addi -10

C Code

int loadsave(int a){

a= a+2;

if(a<5){

a=a+10;

}

a = a+0x8fff;

a = a - 10

}

Assembly Code

Load address

lw a

Since the address is 16 bits while the instruction is 32 bits. There are 16 bits for the immediate which is enough to read any types of the data. “a” is a variable that stores a 16-bits address.

Iteration

Loop:

“content of loop”

beq address Exit

j Loop

Exit:

“exit code”

This is a loop that makes the program goes over and over until the value in the register equals to the value in the specified memory address. When they equal, the program will go to the “Exit” address and finish the loop.

Conditional Statement

beq(bne) address Target

j Else

Target:

“target content”

Else:

“else content”

This is an if statement. If the value of the register equals(not equal) to the data value in the specified address, then executes specified code in the “Target” address. Else, it will executes codes in the “Else” address.

Reading Data from Input Port: Undecided

Reading/Writing from Display Register: Undecided

Writing to Output Port: Undecided

**Euclid’s algorithm**

Lw a:

01010 AAAAAAAAAAAAAAAAAAAAAAAAAAA

Beq 0 Exitb:

01000 0000000000000000 TTTTTTTTTTT

**Loop:**

Lw b:

01010 AAAAAAAAAAAAAAAAAAAAAAAAAAA

Beq 0 Exita:

01000 0000000000000000 TTTTTTTTTTT

Slt a:

00011 AAAAAAAAAAAAAAAA XXXXXX 00011

Beq 1 AMinus:

01000 0000000000000001 TTTTTTTTTTT

Beq 1 BMinus:

01000 0000000000000000 TTTTTTTTTTT

**AMinus:**

Lw a:

01010 AAAAAAAAAAAAAAAAAAAAAAAAAAA

Sub b:

00011 AAAAAAAAAAAAAAAA XXXXXX 00011

Sw a:

01011 AAAAAAAAAAAAAAAAAAAAAAAAAAA

J Loop:

01111 AAAAAAAAAAAAAAAAAAAAAAAAAAA

**BMinus:**

Lw b:

01010 AAAAAAAAAAAAAAAAAAAAAAAAAAA

Sub a:

00011 AAAAAAAAAAAAAAAA XXXXXX 00011

Sw b:

01011 AAAAAAAAAAAAAAAAAAAAAAAAAAA

J loop:

01111 AAAAAAAAAAAAAAAAAAAAAAAAAAA

**Exita:**

Lw a:

01010 AAAAAAAAAAAAAAAAAAAAAAAAAAA

J return:

01111 AAAAAAAAAAAAAAAAAAAAAAAAAAA

**Exitb:**

Lw b:

01010 AAAAAAAAAAAAAAAAAAAAAAAAAAA

J return:

01111 AAAAAAAAAAAAAAAAAAAAAAAAAAA

Load/Store words & Interrupt

Lw a:

01010 AAAAAAAAAAAAAAAAAAAAAAAAAAA

Addi 2:

00100 0000000000000010 XXXXXX 00100

Slt 5:

00011 0000000000000101 XXXXXX 00011

Beq 1 Num:

01000 0000000000000001 TTTTTTTTTTT

Bne 1 Leave:

01001 0000000000000001 TTTTTTTTTTT

Lw :

01010 AAAAAAAAAAAAAAAAAAAAAAAAAAA

Addi 10:

00100 0000000000001010 XXXXXX 00100

Sw a:

01011 AAAAAAAAAAAAAAAAAAAAAAAAAAA

Addi Ox8fff:

00100 0111111111111111 XXXXXX 00100

Addi -10

00100 1111111111110110 XXXXXX 00100