Problem 1:

Assume an architecture with a byte addressable memory. An instruction is stored at location 500 with its address field at location 502. The address field has a value of 300. The location 300 contains the value 140. Registers R1, R2, and R3 contain the values 85, 95, and 105 respectively. What is the effective address of the instruction operand for each of the following addressing modes? Show your work.

a. Direct Memory Addressing mode.
b. Indirect memory Addressing mode.
c. Immediate
d. PC-relative
e. Register Indirect if the location 502 contains the value 2 instead of 300.
f. Indexed assuming that R3 is the index register.

Solution:

Effective address

a. Direct memory addressing mode: EA = 300.
b. Indirect memory addressing mode: EA = 140.
c. Immediate: EA = 502.
d. PC-Relative : EA= 503 + (300x3) = 1403.
e. Register Indirect: EA = 95.
f. Indexed = 105 + 300 = 405.

Problem 2:

Assume an architecture with a byte addressable memory. An instruction is stored at location 300 with its address field at location 301. The address field has a value of 400. An accumulator register R1 contains the number 200. Evaluate effective address if the addressing mode of the instruction is:

a. Direct memory
b. Immediate
c. PC-Relative

d. Register indirect

e. Indexed with R1 as the index register.

Solution:

Effective address

R1 = 200

a. Direct memory: EA = 400

b. Immediate = 301

c. PC-Relative = 302 + (400x2) = 1102

d. Register Indirect = 200

e. Indexed = 200 + 400 = 600

Memory

PC → 300	Opcode
301	400
302	Next instruction

Problem 3:

A two-word instruction is stored in memory at an address designated by the symbol W. The address field of the instruction (stored at W+1) is designated by the symbol Y. The operand used during the execution of the instruction is stored at an address symbolized by Z. An index register contains the value X. State how Z is calculated from the other addresses if the addressing mode of the instruction is:

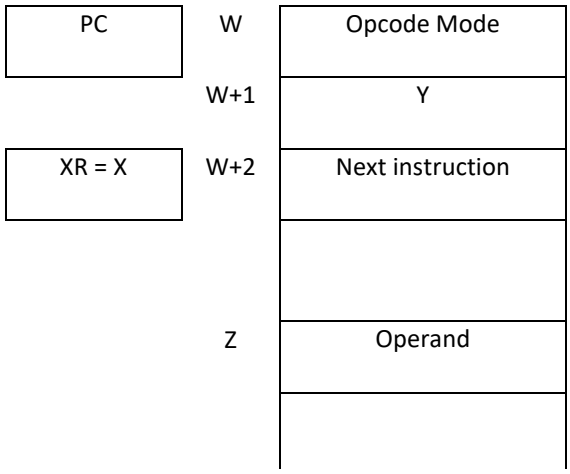
a. Direct memory

b. Immediate

c. PC-Relative

d. Indexed

Solution:



Z =Effective address

a. Direct memory: Z=Y
b. Immediate: Z=W+1
c. PC-Relative: Z=2Y+(W+2)
d. Indexed: Z=X+Y

Problem 4:

A PC-Relative mode branch type of instruction is stored in memory at an address equivalent to decimal 750 (assume one cell per instruction). The branch is made to an address equivalent to decimal 500. What should be the value of the relative address field of the instruction (in decimal)?

Solution:

EA = PC (new) + relative address

Relative address = 500 – 751 = -251

Problem 5:

A PC-Relative mode branch type of instruction is stored in memory at an address equivalent to decimal 1000 (assume two cells per instruction). The branch is made to an address equivalent to decimal 2000. What should be the value of the relative address field of the instruction (in decimal)?

Solution:

EA = PC (new) + (relative addressx2)

2000 = 1002 + (Relative addressx2)

(Relative addressx2) = 2000 - 1002 = 998

Relative address = 998 / 2 = 499

Problem 6:

Consider a hypothetical 24-bit microprocessor having 24-bit instructions composed of two fields: The first byte contains the Opcode and the remainder the operand address. What is the maximum directly addressable memory capacity (in bytes)?

Solution :

Opcode one byte (8 bits) and address 2 Bytes (16 bits).

Therefore, the maximum directly addressable memory capacity (in bytes) → 2^{16} = 64 Kb

Problem 7:

A computer has a 32-bit instructions and 12-bit addresses. If the instructions on this computer are either of type one-address instructions or two-address. If there are 250 two-address instructions, how many one-address instructions can be formulated?

Solution:

8 12 12 = 32 bit

Opcode	Address 1	Address 2

Two address instructions

2^8 = 256 combinations.

256 – 250 = 6 combinations can be used for one address

For one-address instructions, one of the address fields can be used as an extension to the Opcode.

Opcode	Address

One address instructions

6×2^{12}

Maximum number of one address instruction = 6 x 2^{12} = 24,576 instructions

Problem 8:

Consider a hypothetical 32-bit microprocessor having 32-bit instructions composed of two fields: the first byte contains the opcode and the remainder the immediate operand or an operand address.

a) What is the maximum directly addressable capacity (in bytes)?

b) Discuss the impact on the system speed if the microprocessor bus has:

1. A 32 bit local address bus and a 16-bit local data bus.
2. A 16 bit local address bus and a 16-bit local data bus.

c) How many bits are needed for the PC and the IR registers?

Solution:

a) 2^{24} = 16Mbytes

b) 1) If the local address bus is 32 bits, the whole address can be transferred at once and decoded in memory. However, since the data bus is only 16-bits, it will require 2 cycles to fetch a 32-bit instruction or operand.

2) The 16 bits of the address is placed on the address bus can't access the whole memory. Thus a more complex memory interface control is needed to latch the first part of the address and then the second part (since the microprocessor will end in two steps). For a 32-bit address, one may assume the first half will decode to access a "row" in memory, while the second half is sent later to access a "column" in memory. In addition to the two-step address operation, the microprocessor will need 2 cycles to fetch the 32 bit instruction/operand.

c) The program counter must be at least 24 bits. Typically, a 32-bit microprocessor will have a 32-bit external address bus and a 32-bit program counter, unless on-chip segment registers are used that may work with a smaller program counter. If the instruction register is to contain the whole instruction, it will have to be 32-bit long; if it will contain only the opcode (called the opcode register) then it will have to be 8 bits long.

Problem 9:

Discuss briefly the following MIPS addressing modes. State an example for each one of them.

a) Register direct addressing mode

b) PC-relative addressing mode

c) Base-addressing mode

d) Pseudo-direct addressing mode

Solution:

a) ***Register addressing mode:*** In this addressing mode, the instruction holds the number of the register containing the operand. An example of an instruction that uses register direct addressing mode is "add $s0, $s1, $s2", which adds the contents of registers $s1 and $s2 and stores the result in register $s0.

b) ***PC-relative addressing mode:*** In this addressing mode, the program counter (PC) is used to calculate the memory address of the next instruction to be fetched (read) from the memory. It is used with branch instructions. If the branch is taken the value in the instruction is added to the new PC (already incremented by 4). An example of an instruction that uses PC-relative addressing mode is "beq $s1, $s2, label", which branches to the instruction at the label if the contents of registers $s1 and $s2 are equal.

c) ***Base addressing mode***: In this addressing mode, the use of a base register to hold the memory address of the beginning of an array or structure. This mode is used when the operand is located in a memory location that is offset from the base address. An example of an instruction that uses base addressing mode is "lw $t0, 4($s1)", which loads the contents of the memory location at the address $s1+4 into register $t0.

d) ***Pseudo-direct addressing mode***: This is an addressing mode used for jump instructions. This mode is used when the target address of the jump instruction is too far away to be represented by the 16-bit offset field in the instruction. An example of an instruction that uses pseudo-direct addressing mode is "j label", which jumps to the instruction at the label. The target address of the jump instruction is calculated by concatenating the upper 4 bits of the PC with the 26-bit address field in the instruction.

Problem 10:

The MIPS architecture supports some and doesn't support some of the addressing modes below. Synthesize these instructions by writing one or more MIPS lines that do the same functionality.

a) Load $s1, ($s2) // Register Indirect addressing mode.

a) Load $s1, @($s2) // Memory Indirect addressing mode, assuming that $S2 contains the first memory address

b) Load $s1, ($s2+$s3+200) // indexed with displacement addressing mode.

c) Load $s1, (0x1122AABB) // memory direct addressing mode.

Solution:

a) Load $s1, ($s2) // Register Indirect addressing mode.

This could be done by the following MIPS instructions:

```
lw $t2, 0($s2)          // now $t2 has the address.
lw $s1, 0($t2)          // now $s1 has the operand.
```

b) Load $s1, @($s2) // Memory Indirect addressing mode, assuming that $S2 contains the first memory address

This could be done by the following MIPS instructions:

```
lw $t2, 0($s2)          // now $t2 has the first memory address.
lw $t3, 0($t2)          // now $t3 has the second memory address.
lw $s1, 0($t3)          // now $s1 contains the operand.
```

c) Load $s1, ($s2+$s3+200) // indexed with displacement addressing mode.

This could be done by the following MIPS instructions:

```
add $t1, $s2, $s3
addi $t1,200
lw $s1, 0($t1)          // now $s1 contains the operand.
```

d) Load $s1, (0x1122AABB) // memory direct addressing mode.

```
li $t1, 0x1122AABB
lw $s1, 0($t1)
```

Problem 11:

Assume in the MIPS PC register has the value of 0x1122AABB, a j instruction has an address field of 0x003AABB. What is the address that the jump instruction would like to reach?

Solution:

After adding the 00 and concatenating the upper four significant bits from PC, the address that the jump instruction wants to reach is 0x100EAAEC.

Problem 12:

Given the following MIPS code, state the addressing mode in each of its lines:

```
loop: addi  $s1, $s2, 60
      add $s3, $s1, $t0
      lw $t4, 4($s3)
      j loop
```

Solution:

```
loop: addi  $s1, $s2, 60        // Immediate addressing mode.
      add $s3, $s1, $t0         // Register direct addressing mode.
      lw $t4, 4($s3)            // base addressing mode.
      j loop                    // pseudo-direct addressing mode.
```