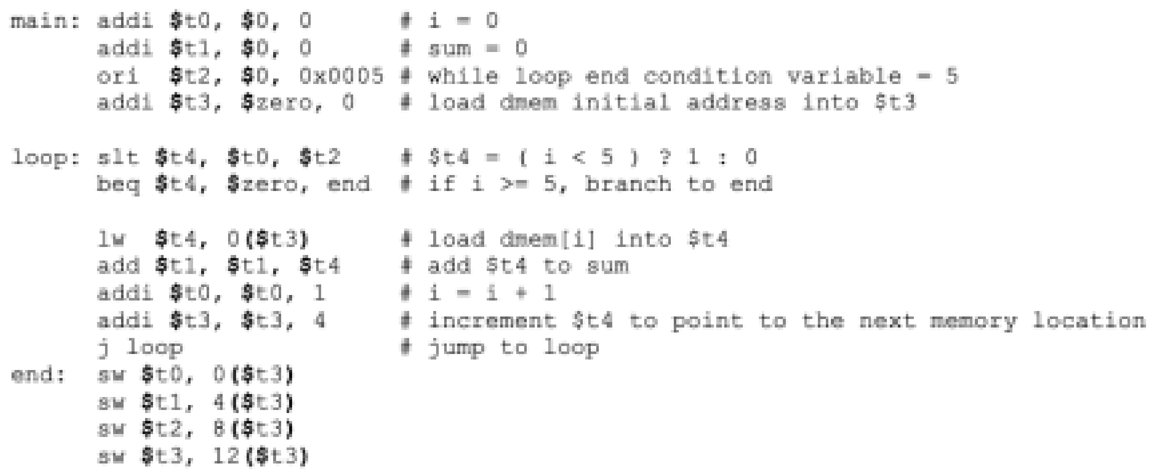
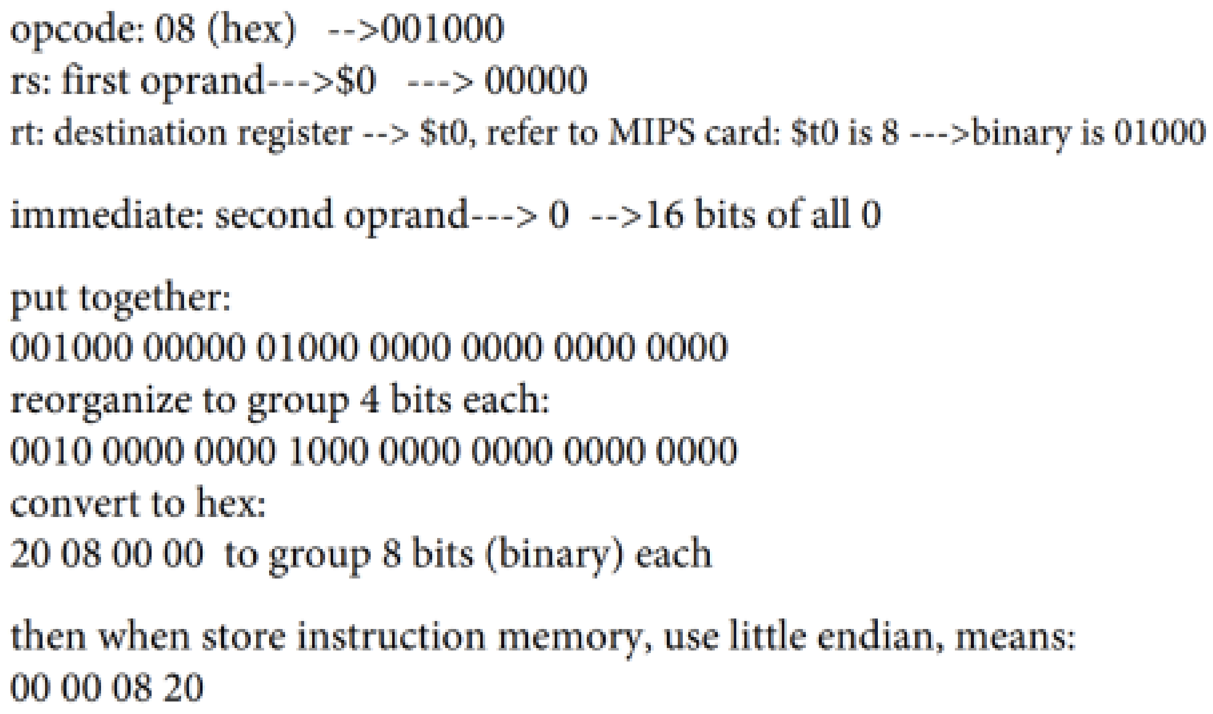


```
main: addi $t0, $0, 0       # i = 0
      addi $t1, $0, 0       # sum = 0
      ori  $t2, $0, 0x0005  # while loop end condition variable = 5
      addi $t3, $zero, 0    # load dmem initial address into $t3

loop: slt $t4, $t0, $t2     # $t4 = ( i < 5 ) ? 1 : 0
      beq $t4, $zero, end   # if i >= 5, branch to end

      lw  $t4, 0($t3)       # load dmem[i] into $t4
      add $t1, $t1, $t4     # add $t4 to sum
      addi $t0, $t0, 1      # i = i + 1
      addi $t3, $t3, 4      # increment $t4 to point to the next memory location
      j loop                # jump to loop
end:  sw $t0, 0($t3)
      sw $t1, 4($t3)
      sw $t2, 8($t3)
      sw $t3, 12($t3)
```

addi $t0,$0,0



opcode: 08 (hex)   -->001000
rs: first oprand--->$0   ---> 00000
rt: destination register --> $t0, refer to MIPS card: $t0 is 8 --->binary is 01000

immediate: second oprand---> 0 -->16 bits of all 0

put together:
001000 00000 01000 0000 0000 0000 0000
reorganize to group 4 bits each:
0010 0000 0000 1000 0000 0000 0000 0000
convert to hex:
20 08 00 00 to group 8 bits (binary) each

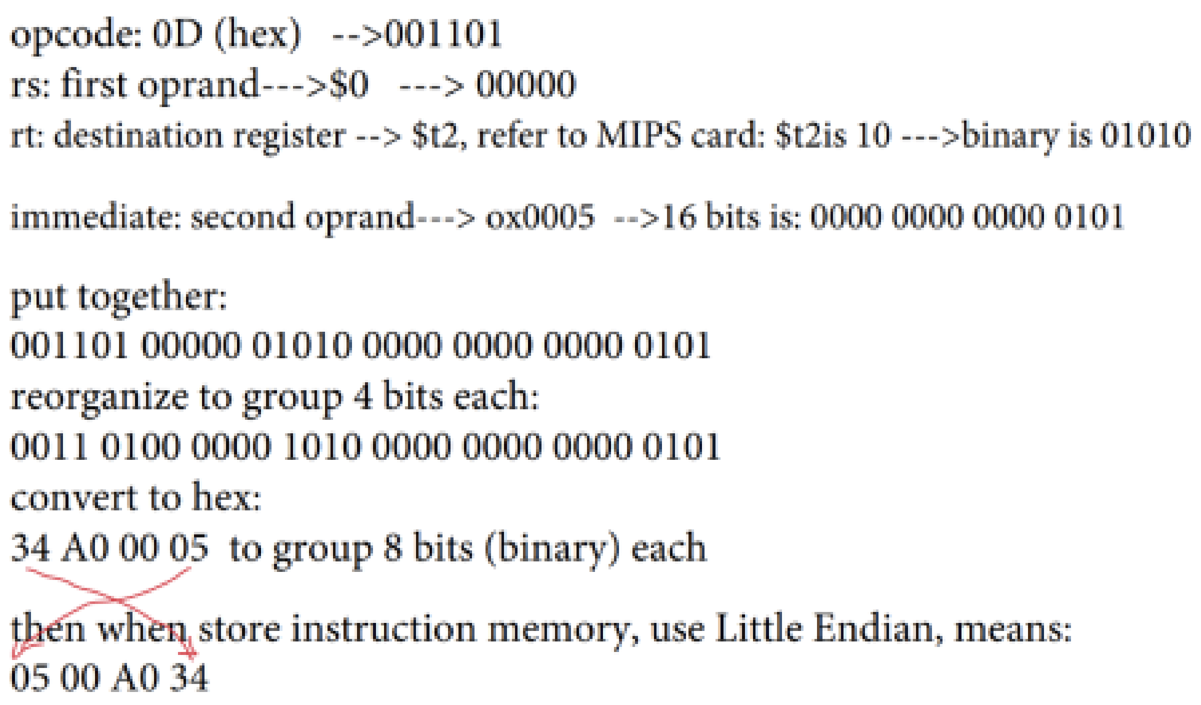
then when store instruction memory, use little endian, means:
00 00 08 20

addi $t1,$0,0

```
Opcode: 08(hex) -> 001000
Rs: first operand -> $0 -> 00000
Rt: destination register -> $t1 ->binary is 01001
Immediate: second operand -> 0 - > 16 bits of 0's -> 0000 0000 0000
0000
0010 0000 0000 1001 0000 0000 0000 0000
Convert to hex:
20 09 00 00
```

```
Store instruction memory:
00 00 09 20
```

`orri $t2, $0, 0x0005`



opcode: 0D (hex) -->001101
rs: first oprand--->$0 ---> 00000
rt: destination register --> $t2, refer to MIPS card: $t2is 10 --->binary is 01010

immediate: second oprand---> ox0005 -->16 bits is: 0000 0000 0000 0101

put together:
001101 00000 01010 0000 0000 0000 0101
reorganize to group 4 bits each:
0011 0100 0000 1010 0000 0000 0000 0101
convert to hex:
34 A0 00 05 to group 8 bits (binary) each

then when store instruction memory, use Little Endian, means:
05 00 A0 34

`addi $t3, $zero, 0`

```
Opcode: 8(hex) -> 001000
Rs: first operand -> $zero -> 00000
Rt: destination register -> $t3 ->binary is 01011
Immediate: second operand -> 0 - > 16 bits of 0’s -> 0000 0000 0000
0000
0010 0000 0000 1011 0000 0000 0000 0000
Convert to hex:
20 0B 00 00
Store instruction memory:
00 00 0B 20
```

`slt $t4, $t0, $t2`

SLT $destination register's address, $first source register's address, $second source register's address

```
Opcode: 000000
Rs: $t0 -> 01000
Rt: $t2 -> 01010
```

```
Rd: $t4 -> 01100
Shamt: 00000
Funct: 101010
0000 0001 0000 1010 0110 0000 0010 1010
01 0A 60 2A
2A 60 0A 01
```

`beq $t4, $zero, end = 5`

BEQ $first source register's address, $second source register's address, branch value

```
Opcode: 000100
Rs : $t4 -> 01100
Rt : $zero -> 00000
Branch(offset): 0000 0000 0000 0101
0001 0001 1000 0000 0000 0000 0000 0101
11 80 00 05
05 00 80 11
```

`lw $t4, 0($t3)`
LW $destination register's address, offset($source register's address).

```
Opcode: 100011
Rs: $t3 -> 01011
Rd: $t4 -> 01100
Offset : 0 -> 0000 0000 0000 0000
1000 1101 0110 1100 0000 0000 0000 0000
8D 6C 00 00
00 00 6C 8D
```

`add $t1, $t1, $t4`
ADD $destination register's address, $first source register's address, $second source register's address.

```
Opcode : 000000
Rs: $t1 -> 01001
Rt: $t4 -> 01100
Rd: $t1 -> 01001
Shamt: 00000
```

```
Funct: 100000
0000 0001 0010 1100 0100 1000 0010 0000
01 2C 48 20
20 48 2C 01
```

addi $t0, $t0, 1

ADDI $destination register's address, $source register's address, immediate data.

```
Opcode: 001000
Rs: $t0 ->01000
Rd: $t0 -> 01000
Immediate: 1-> 0000 0000 0000 0001

0010 0001 0000 1000 0000 0000 0000 0001
21 08 00 01
01 00 08 21
```

addi $t3, $t3, 4

ADDI $destination register's address, $source register's address, immediate data.

```
Opcode: 001000
Rs: $t3 -> 01011
Rd: $t3 -> 01011
Immediate: 4 -> 0000 0000 0000 0100
0010 0001 0110 1011 0000 0000 0000 0100
21 6B 00 04
04 00 6B 21
```

J loop

```
Opcode: 000010
Offset: 0000 0000 0000 0000 0000 0001 00
0000 1000 0000 0000 0000 0000 0000 0100
08 00 00 04
04 00 00 08
```

sw $t0, 0($t3)

SW $source register's address, offset($destination register's address).

```
Opcode: 101011
Rd: $t3 -> 01011
Rs: $t0 -> 01000
Offset: 0-> 0000 0000 0000 0000
```

```
1010 1101 0110 1000 0000 0000 0000 0000
AD 68 00 00
00 00 68 AD
```

sw $t1, 4($t3)

SW $source register's address, offset($destination register's address).

```
Opcode: 101011
Rd: $t3 -> 01011
Rs: $t1 -> 01001
Offset: 4-> 0000000000000100

1010 1101 0110 1001 0000 0000 0000 0100
AD 69 00 04
04 00 69 AD
```

sw $t2, 8($t3)

SW $source register's address, offset($destination register's address).

```
Opcode: 101011
Rd: $t3 -> 01011
Rs: $t2 -> 01010
Offset: 8-> 0000000000001000

1010 1101 0110 1010 0000 0000 0000 1000
AD 6A 00 08
08 00 6A AD
```

sw $t3, 12($t3)

SW $source register's address, offset($destination register's address).

```
Opcode: 101011
Rd: $t3 -> 01011
Rs: $t3 -> 01011
Offset: 12-> 0000000000001100

1010 1101 0110 1011 0000 0000 0000 1100
AD 6B 00 0C
0C 00 6B AD
```