Jose Franco Baquera

March 4, 2018

CS - 273

Homework #6

1. **The CALL and RET instruction.**

   a. **How many bytes do CALL and RET instructions each have?**

      Each CALL instruction will have 4 bytes.

      Each RET instruction will have 2 bytes.

   b. **With each CALL instruction, is the stack pointer incremented or decremented, and by how much?**

      After each CALL instruction the stack pointer is decremented by 2.

   c. **With each RET instruction, is the stack pointer incremented or decremented, and by how much?**

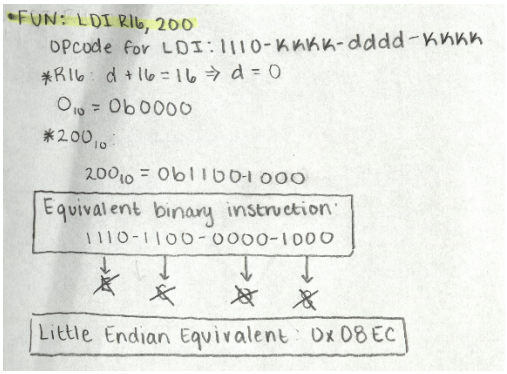
      After each RET instruction the stack pointer is incremented by 2.

2. **Convert the following assembly code to binary by hand using the instruction table in the end of the 273 booklet. You must use the little endian in representing the low and high bytes at each program memory location.**

   (NOTE: We assume that the first instruction is at memory location 1, the second instruction is at memory location 2, etc.)

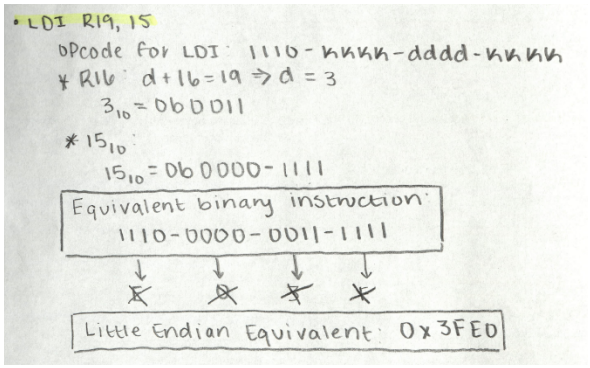
(1) FUN: LDI R16, 200 → `1110-1100-0000-1000` → "Little Endian" 0x08EC

Proof:



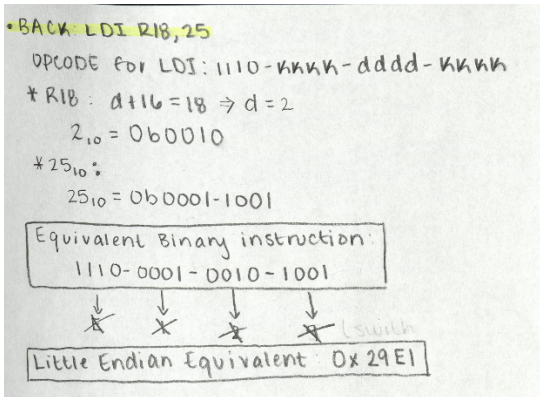
(2) LDI R19, 15 → 1110-0000-0011-1111 → "Little Endian" 0x3FE0

Proof:



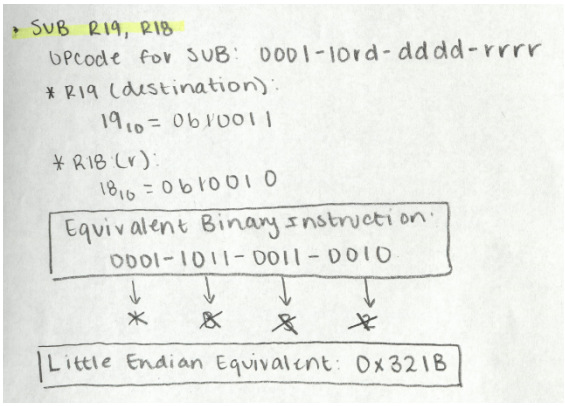
(3) BACK: LDI R18, 25 → 1110-0001-0010-1001 → "Little Endian" 0x29E1

Proof:



(4) SUB R19, R18 → 0001-1011-0011-0010 → "Little Endian" 0x321B

Proof:

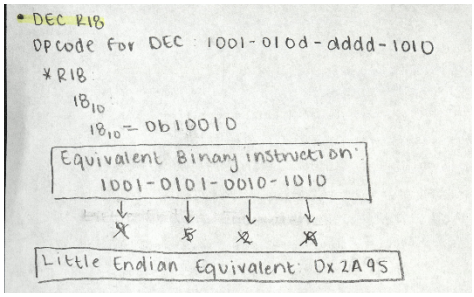


(5) HERE: NOP → 0000-0000-0000-0000 → "Little Endian" 0x0000

Proof: No proof is necessary for this instruction since it's straightforward.

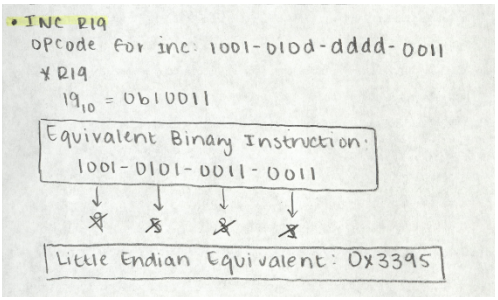
(6) DEC R18 → 1001-0101-0010-1010 → "Little Endian" 0x2A95

Proof:



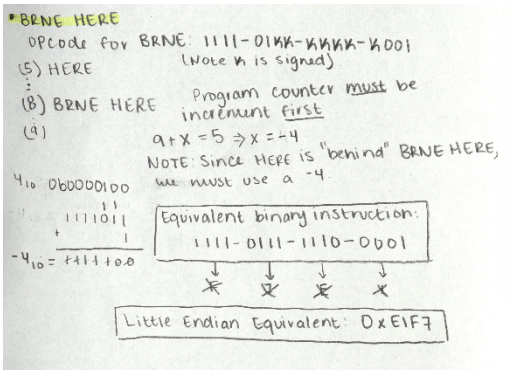
(7) INC R19 → 1001-0101-0011-0011 → "Little Endian" 0x3395

Proof:



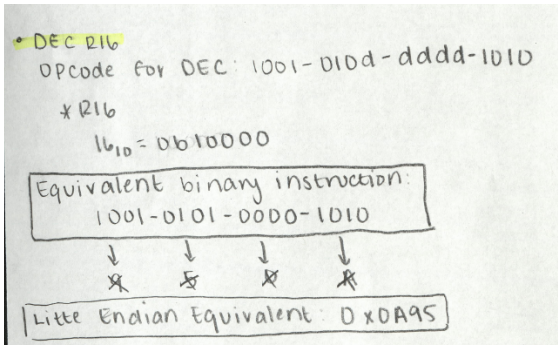
(8) BRNE HERE → 1111-0111-1110-0001 → "Little Endian" 0xE1F7

Proof: (NOTE: Offset is -4)



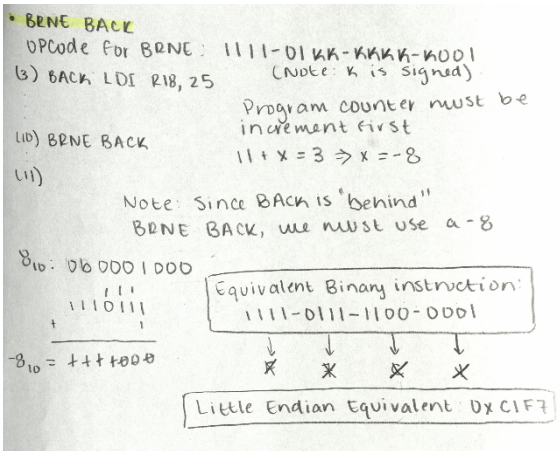
(9) DEC R16 → 1001-0101-0000-1010 → "Little Endian" 0x0A95

Proof:



(10) BRNE BACK → 1111-0111-1100-0001 → "Little Endian" 0xC1F7

Proof: (NOTE: Offset is -8)



(11) RET → 1001-0101-0000-1000 → "Little Endian" 0x0895

Proof: No proof is necessary for this instruction since it's straightforward.

3. **Find the time delay of the above program if the system has an AVR with a frequency of 16MHz.**

FUN: LDI R16, 200 → 1 cycle = **1 cycle**

LDI R19, 15 → 1 cycle = **1 cycle**

BACK: LDI R18, 25 → 1 cycle * 200 = **200 cycles**

SUB R19, R18 → 1 cycle * 200 = **200 cycles**

HERE: NOP → (1 cycle * 10) * 1 + (1 cycle * 25) * 199 = **4985 cycles**

DEC R18 → (1 cycle * 10) * 1 + (1 cycle * 25) * 199 = **4985 cycles**

INC R19 → (1 cycle * 10) * 1 + (1 cycle * 25) * 199 = **4985 cycles**

BRNE HERE → ((2 cycles * 9) + 1 cycle) * 1 + ((2 cycles * 24) + 1 cycle) * 199 = **9770 cycles**

DEC R16 → 1 cycle * 200 = **200 cycles**

BRNE BACK → (2 cycles * 199) + 1 cycle = **399 cycles**

RET → 4 cycles = **4 cycles**

(NOTE: 1Hz is one cycle per second. In addition, BRNE uses 2 cycles if it does not fall through while 1 cycle if it falls through.)

- Total Number of Cycles = 25,730 cycles
- Frequency = 16MHz = 1/(16*10^6)sec = 1/16,000,000 sec
- **ANSWER:**

  Total Time Delay = Frequency * Cycles = (1/16,000,000) * (25,730)

  = 0.001608125 seconds = 1608.125 micro seconds = 1.608125 milliseconds (ms)