CS2323: Computer Architecture, Autumn 2024

Homework-2: Conditionals and Memory access

Ahmik Virani
ES22BTECH11001

Q1)
Idea : Add the value in register x5 a total of value of x6 times.
That is :

```
int counter = *(x6);
int answer = 0;
for(int i = 0 ; i < counter ; i++)
{
   answer += *(x5);
}
return answer; (store answer in register x10)
```

```
  addi x20, x0, 0          //initialize a counter variable in register x20
  addi x10, x0, 0          //This will store the final answer, initialize it to 0
LOOP:
  beq x20, x6, EXIT        //If counter reaches the value of x6, then branch to EXIT
  addi x20, x20, 1         //Increase the counter by 1
  add x10, x10, x5         //Add the value of x5
  beq x0, x0 LOOP          //Go back to the label LOOP (unconditional jump)
EXIT:                      //Exit
  addi, x10, x10, 0
```

Q2)

```
add x18, x14, x0                //I am using x18 as temporary variable to store address of A

LOOP:

   bge x12, x11, EXIT           //if q>=p exit the loop
   addi x7, x13, 0              //First store the value of i in register x7
   slli x7, x7, 3               //Multiply the value by 8, (considering A[i] is long word)
   add x18, x14, x7             //Add 8 * i to address of A to get address of A[i]
   ld x20, 0(x18)               //Load the value of A[i] to register x20
   beq x20, x0, EXIT            //if A[i] == 0, exit the loop
```

```
   blt x0, x20, ELSE          //if A[i] > 0, go to else condition
   add x11, x11, x20          //p = p + A[i] if A[i] <= 0
   beq x0, x0, NEXT           //Once inside if condition, do not execute the else condition

 ELSE:
    add x11, x0, x20          // p = A[i] if A[i] > 0
 NEXT:
    addi x13, x13, 2          //Add the value 2 to i


    beq x0, x0, LOOP          //Unconditional jump to the start of the loop label again

EXIT:
        addi x0, x0, 0        //Exit the loop
```

Q3)

1. lhu x3, 0(x1)
   Here the value is 0x0000000000003939 because half word is 2 bytes and we will take the least significant 2 bytes that is 4 digits in hexadecimal
2. lh x3, 0(x1)
   Here also the value will be 0x0000000000003939 because the most significant bit is 3 which is less than 8, so will be positive and the initial bits will be 0's

3. lh x3, 2(x1)
   Here the value will be 0xffffffffffff9393 because the most significant bit is 9 which is more than 8, so will be negative and the initial bits will be 1's, and we are starting at offset of 2 bytes so we will leave first 4 digits in hexadecimal and halfword so take next 2 bytes (4 digits).

4. ld x3, 0(x1)
   The value will be 0xa55aa5a593933939 itself because it is already represented as a double word.
5. lw x3, 12(x1)
   The value will be 0x0000000039933939 because we are going at an offset of 12 bytes that is 24 digits of hexadecimal. One long word has 64 bytes that is 16 digits so we will jump to the next word, but still need to traverse 8 more digits and then store the word (4 bytes)
6. lbu x3, 7(x1)
   Now we are just loading 1 byte at an offset of 7 bytes, so we will take 0x00000000000000a5 and the initial bits are zero because the value is unsigned
7. lb x3, 7(x1)
   Now we are just loading 1 byte at an offset of 7 bytes, so we will take

0xffffffffffffffa5 and the initial bits are f because the value is signed, and value of ‘a’ in hexadecimal is more than 8

8. lb x3, 6(x1)

   Now we are just loading 1 byte at an offset of 6 bytes, so we will take 0x000000000000005a and the initial bits are zero because the value 5 < 8.

9. ld x3, 3(x1)

   Here we get 0x5aa5a5a55aa5a593 because we take double word that is 8 bytes starting from an offset of 3 bytes of the first double word, so to take 8 bytes we need to even take some from the next double word. (They are at contiguous locations)

10. lwu x3, 6(x1)

    Here we get 0x00000000a5a5a55a because it is unsigned so leading bits are zero, and we take word i.e. 4 bytes from an offset of 6 bytes of x1