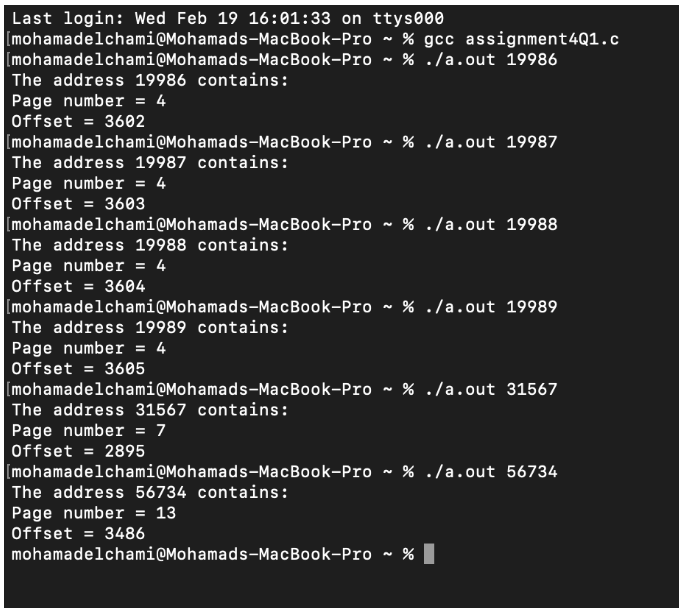
Assignment 4

"I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work." - Mohamad Elchami [redacted]

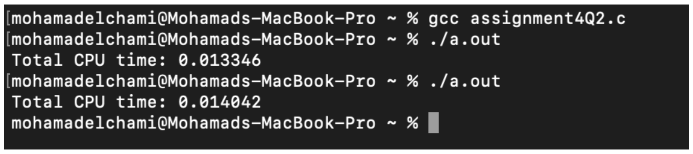
Question 1:



```
Last login: Wed Feb 19 16:01:33 on ttys000
[mohamadelchami@Mohamads-MacBook-Pro ~ % gcc assignment4Q1.c
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 19986
The address 19986 contains:
Page number = 4
Offset = 3602
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 19987
The address 19987 contains:
Page number = 4
Offset = 3603
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 19988
The address 19988 contains:
Page number = 4
Offset = 3604
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 19989
The address 19989 contains:
Page number = 4
Offset = 3605
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 31567
The address 31567 contains:
Page number = 7
Offset = 2895
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 56734
The address 56734 contains:
Page number = 13
Offset = 3486
mohamadelchami@Mohamads-MacBook-Pro ~ %
```

Run of part one

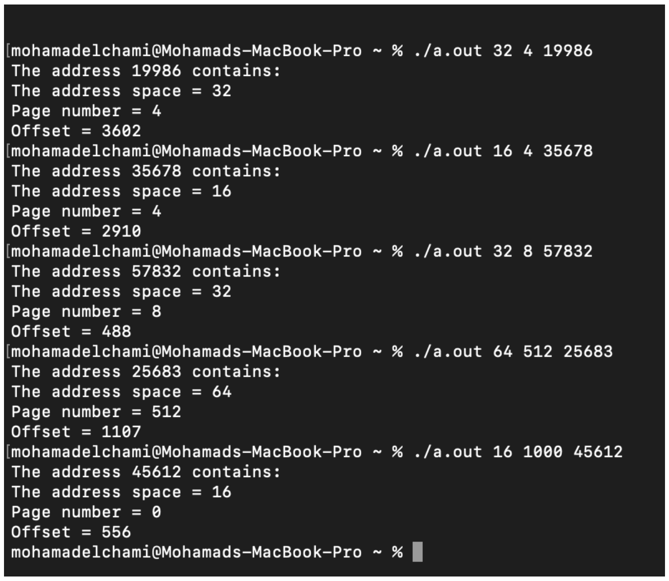
Part one was implemented quite simply. The program takes the address from as input from standard input or terminal in this case. The argument is converted to an unsigned long integer via atoll(). The page is then calculated using the left shift operator (>>) to shift the binary to the left by 12. The offset is then calculated using the AND operator.



```
[mohamadelchami@Mohamads-MacBook-Pro ~ % gcc assignment4Q2.c
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out
Total CPU time: 0.013346
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out
Total CPU time: 0.014042
mohamadelchami@Mohamads-MacBook-Pro ~ %
```

Run of part 2

Part 2 did not take a command line inputs, it generated a random address between 0 and 2^{32}-1. To achieve this, I created a min and max variable to hold those values. I also created a variable n to control loop termination to have the loop tun 1000000 times, as per the requirements. Before the loop begins the time is captured. The loop is run, each iteration the address is assigned a random address, then page and offset are then calculated. Once the loop it terminated the time is then captured again, the difference is calculated and printed onto the screen. The CPU runtime is extremely fast for the amount of iterations the loop actually completes.



```
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 32 4 19986
 The address 19986 contains:
 The address space = 32
 Page number = 4
 Offset = 3602
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 16 4 35678
 The address 35678 contains:
 The address space = 16
 Page number = 4
 Offset = 2910
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 32 8 57832
 The address 57832 contains:
 The address space = 32
 Page number = 8
 Offset = 488
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 64 512 25683
 The address 25683 contains:
 The address space = 64
 Page number = 512
 Offset = 1107
[mohamadelchami@Mohamads-MacBook-Pro ~ % ./a.out 16 1000 45612
 The address 45612 contains:
 The address space = 16
 Page number = 0
 Offset = 556
 mohamadelchami@Mohamads-MacBook-Pro ~ %
```

Run of part three

Part three take many more arguments than part one. I added the variable for Address Space. We then move to confirming that address space and the page size, of in this case I left it as page number.

First, I check the address space by using and if statement passing the argument to a function that checks if it is equal to 16, 32, or 64, check_address(). If it is the value is assigned to the variable, if not then it results in 0.

Second, I check the page size to be a power of 2 by using the same method above but using a unction that checks the power, check_power(). If the value is a power of 2 then the value is assigned to the variable, if it is not then it results in 0.

Lastly, the offset is calculated and the values are printed.