Results

- Q1: 14, a*b- a right shift 1
- Q2: 432, finding the biggest number
- Q3: 153,370,371,407, finding the number which has the attribute that its cubic of hundreds digit, cubic of tens digit,cubic of units digit add together which equal to itself.
- Q4: compare and sub(less)/add(bigger).

1 Q1

- PUSH EBP
  Push the value of the EBP into the stack to protect the esp pointer.
- MOV EBP, ESP
  Set the EBP to point to the top of the stack, which is the value of ESP. This step is to lock down the position of address of the stack. Which also means that it is establishing a new stack frame .
- AND ESP, 0FFFFFFFF0H
  Align the ESP with the 16 bit address,aligns the stack defaulting to 16-byte alignment.
- SUB ESP, 20H
  Let the pointer to the top of stack go downwards to reserve a 20h address for the program.
- call _main
  Call the main function.
- MOV DWORD PTR[esp+1CH], 3
  Store 3 into the address of [esp+1ch] which is [esp+28] in the stack.
- MOV Dword ptr [esp+18h], 5
  Store the 5 into the next 4 bit address which is esp+24.
- MOV Dword ptr [esp+14h], 0
  Store the 0 into the next 4 bit address in the stack which is esp+20.
- MOV Eax, [esp+1Ch]
  Put the value of 3 into the 32bit register eax.
- IMUL Eax, [esp+18h]
  Multiply the 3 with the 5 and stire the result 15 into the register eax, because no overflow, so the flag does not change.
- MOV Edx, eax
  Store the value of 15 into the register edx.

- MOV Eax, [esp+1Ch]
  Store the value of 3 into the register eax.
- MOV Ecx, eax
  Move the value of eax which is 3 into the register ecx, ecx = eax = 3.
- SHR Ecx, 1Fh
  Right shift the binary of ecx by 32, which means to cut off the sign bit of ecx, on this case, it means 0 because ecx is a positive value.
- ADD Eax, ecx
  Add the value in the register ecx with the eax which means eax = eax+ecx = 3+0 = 3 = 11.
- SAR Eax, 1
  Shift arithmetic right register eax by 1 which means eax = 1 for now.
- SUB Edx, eax
  Subtraction calculation edx = esx - eax = 15 - 1 = 14.
- MOV Eax, edx
  Move the value of 14 into the register eax.
- MOV [esp+14h], eax
  MOve the value of 14 into the stack at the address of esp=14 from the top.
- MOV Eax, [esp+14h]
  MOve back the value.
- MOV [esp+4], eax
  Set the value of 14 into the address of 4 from the top of stack. MOV Dword ptr [esp], offset aD ; "%d"
  Set the entrance of the next call function offset to address 4 from the top of stack.
- Call _printf
  Call the function of print
- MOV Eax, 0
  Set eax as 0.

  END OF THE PROGRAM

2 Q2

The start of the program is the same as the Q1, we can refer to Q1 for the start of the analysis.

- mov dword ptr [esp+18h], 0Ch
  Set the address of 24 from the top of stack as the value of 0Ch, which is 12.

- mov dword ptr [esp+1Ch], 0Fh
  Set the address of 28 from the top of stack as the value of 15.

- mov dword ptr [esp+20h], 0DDh
  Set the address of 32 from the top of stack as the value of 221.

- mov dword ptr [esp+24h], 3
  Set the address of 36 from the top of stack as the value of 3.

- mov dword ptr [esp+28h], 1B0h
  Set the address of 40 from the top of stack as the value of 432.

- mov dword ptr [esp+2Ch], 36h
  Set the address of 44 from the top of stack as the value of 54.

- mov dword ptr [esp+30h], 10h
  Set the address of 48 from the top of the stack as the value of 16.

- mov dword ptr [esp+34h], 43h
  Set the address of 42 from the top of the stack as the value of 67.

- mov dword ptr [esp+3Ch], 0
  Set the address of 46 from the top of the stack as the value of 0.

- mov dword ptr [esp+38h], 0
  Set the address of 50 from the top of the stack as the value of 0.

- jmp short loc_40157F
  Jump to the location of 40157F to execute the code there.
  After that, it will jump and not execute in order.

From there, I will analysis the detailed code with jump of procedure and by the executing order

- loc_40157F:
  This is the location in the stack.Which is the next location of the code.

- cmp dword ptr [esp+38h], 7
  Compare the value in the address of 38h from the top of stack with the 7, which means that compare 0 with 7 at the first time, so we got a less than result and set CF =1 ,SF= 1,OF =0.

- jle short loc_401560
  Because at the first time, SF!=OF, so the jump condition has been satisfied, also, jump to the location 401560. It is a less or equal trigger.

- loc_401560:
  The jumping location of code.

- mov eax, [esp+38h]
  Move the value of 38h to the eax register, at the first time, eax = 0.

- mov eax, [esp+eax*4+18h]
  At the first time, eax = 0, esp+eax*4+18h = 1Ch , so eax = ptr [esp+18h] = 12.

- cmp eax, [esp+3Ch]
  Compare the value of eax = 12 ,with the value in ptr [esp+3Ch] = 0, so is a bigger than.

- jle short loc_40157A
  Not triggered, ignore. Once it has been triggered, it means that the current value is less than the value in the [esp+3Ch] and it will jump to another iteration.

- mov eax, [esp+38h]
  Move the value of [esp+38h] into the eax, eax = 0.

- mov eax, [esp+eax*4+18h]
  Store the value of [esp+eax*4+18h] which is [esp+18h] in the eax, eax = 12.

- mov [esp+3Ch], eax
  Store the value in the eax into the [esp+3Ch], at the first time, it is [esp+3Ch] = 12.

- loc_40157A:
  This is the location in the stack and other line of code will call , such as line: _main+6C would call this line.

- add dword ptr [esp+38h], 1
  Add 1 to the address 38h from the top of stack, so the value in this address should be 0+1 = 1, but later, it will iterate more times until [esp+38h] = 8, because other line of code will call this line.

After this code, the code would go down to the code in loc_40157F which I have explained at the very beginning, and after that, the code would iterate until the ptr [esp+38h] is larger than 7, we can find that in every iteration, the code compare the current pointer with the value stored in [esp+3Ch], if less, skip, if bigger, replace it.

- mov eax, [esp+3Ch]
  Move the value in the [esp+3Ch] into the register eax, which is eax = 1B0h = 432.

- mov [esp+4], eax
  Write the eax = 432 into the [esp+4].

- mov dword ptr [esp], offset aD ; "%d"
  Set the entrance of the next call function offset to address 4 from the top of stack.

- Call _printf
  Call the function of print.

432
END OF PROGRAM

3 Q3

- call _main
  Start of the program.

- mov dword ptr [esp+1Ch], 64h
  Store the value of 64h in the 1Ch from the top of stack.

- jmp loc_4015D6
  Jump to the code section of stack : 4015D6.

From there, I will analysis the detailed code with jump of procedure and by the executing order

- loc_4015D6:
  The location of the next executing program.

- cmp dword ptr [esp+1Ch], 3E7h
  Compare the value in the address [esp+1Ch] with the 3E7h, which is at the first time, compare the value of 100 with 999.

- jle loc_40151B
  IF low or equal with 3E7h, the jump would be triggered, and jump to 40151B in the stack. Otherwise, ignore.

Jumping to 40151B

- mov eax, [esp+1Ch]
  Store the value of [esp+1Ch] into the register EAX = 100.

- mov edx, 51EB851Fh
  Store the value of 51EB851F = 1374389535 = binary(10100011110101110000101000011111), which is a 31 bit into the 32 bit register, so the uppermost bit is 0.

- mov eax, ecx
  Store the value into register eax, eax = ecx = 100.

- imul edx
  Multiply the value in edx with eax, and store the value into edx:eax
- sar edx, 5
  Shift arithmetic right the edx by 5.
- mov eax, ecx
  Store eax = ecx = 100.
- sar eax, 1Fh
  Get the sign bit of the eax which is 0, eax = 0.
- sub edx, eax
  Edx = edx-eax.
- mov eax, edx
  Store eax = edx.
- mov [esp+18h], eax
- mov eax, [esp+18h]
  Exchange the value in these address.
- imul edx, eax, -64h
  For edx = eax , edx = eax * -64 h in the register.
- mov eax, [esp+1Ch]
  For eax = [esp+1Ch].
- lea ecx, [edx+eax]
  Set the value in the edx+eax into the ecx, which is let ecx in the register.
- mov edx, 66666667h
  Store edx = 66666667h.
- mov eax, ecx
  For eax = ecx.
- imul edx
  For eax = edx*eax which is edx:eax in register and high 32 store with low 32, edx with high bit.
- sar edx , 2
  Right shit by 2.
- mov eax, ecx
  For eax = ecx
- sar eax, 1Fh
  Right shit to get the bit sign of eax = 1.

- sub edx, eax
  For edx = edx - eax
- mov [esp+14h], eax
- mov ecx, [esp+1Ch]
  For ecx = 100.
- next lines :
  For edx = 66666667h, eax = ecx = 100, edx:eax = edx * eax , edx = 128, eax = 28. Then right shit edx by 2. Sign bit eax = 0.
- sub edx, eax
  Edx = edx - 0 = edx.
- mov eax, edx
  Eax = edx .
- shl eax, 2
  Left shit eax by 2. Eax =128.
- add eax, edx
  Eax = eax + edx
- add eax, eax
  Eax = 2*eax.
- sub ecx, eax
  Ecx = ecx -eax .
- mov eax, ecx
  For eax = ecx.
- for next lines:
  Put [esp+10h] . eax = [esp+18h] . eax = [esp+18h]$\hat{3}$. eax = [esp+18h]$\hat{3}$ + [esp+14h]$\hat{3}$. eax = [esp+18h]$\hat{3}$ + [esp+14h]$\hat{3}$ + [esp+10h]$\hat{3}$.
- cmp eax, [esp+1Ch]
  Compare the eax with 100.
- jnz short loc_4015D1
  Jump if not zero, or not equal.
- next code :
  Next lines of code is a count down code to count how many times the code iterates.

  The code is mainly a 4 steps of variable manipulation in the stack
  Location of stack from 40151B to 401534: The code is to calculate the value of extracting the hundreds digit of the number

Location of stack from 401538 to 40155B is to extract the tens digit of the number
Location of stack from 40155F to 401583 is to extract the units digit of the number
Location of stack from 401587 to 40158B5 is to calculate the cubic of three numbers extracted before and accumulated together
The last code is to iterate until them find a number which has the attribute that its cubic of hundreds digit, cubic of tens digit,cubic of units digit add together and equal to itself. The result would be 153,370,371,407

4 Q4

Attached code.