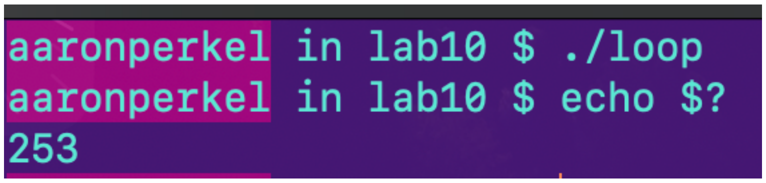
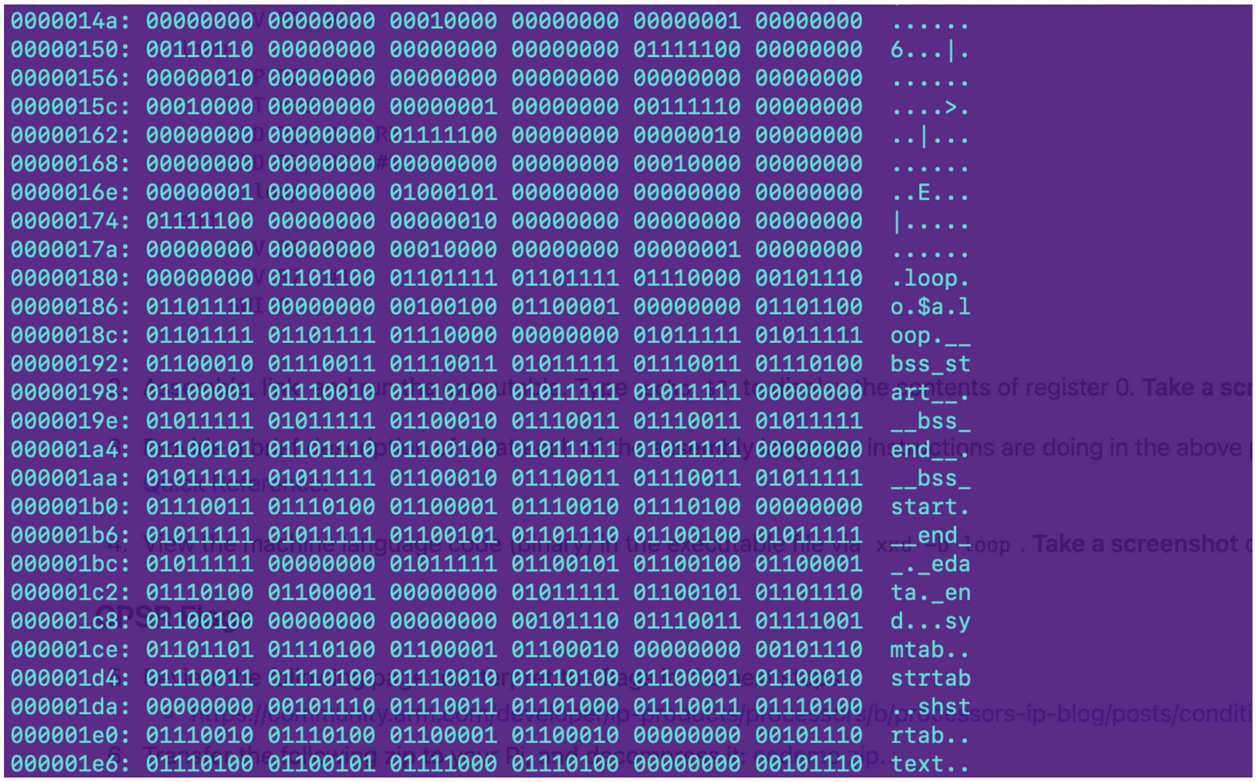
Aaron Perkel
CS2210
12/01/23
Lab 10



```
aaronperkel in lab10 $ ./loop
aaronperkel in lab10 $ echo $?
253
```

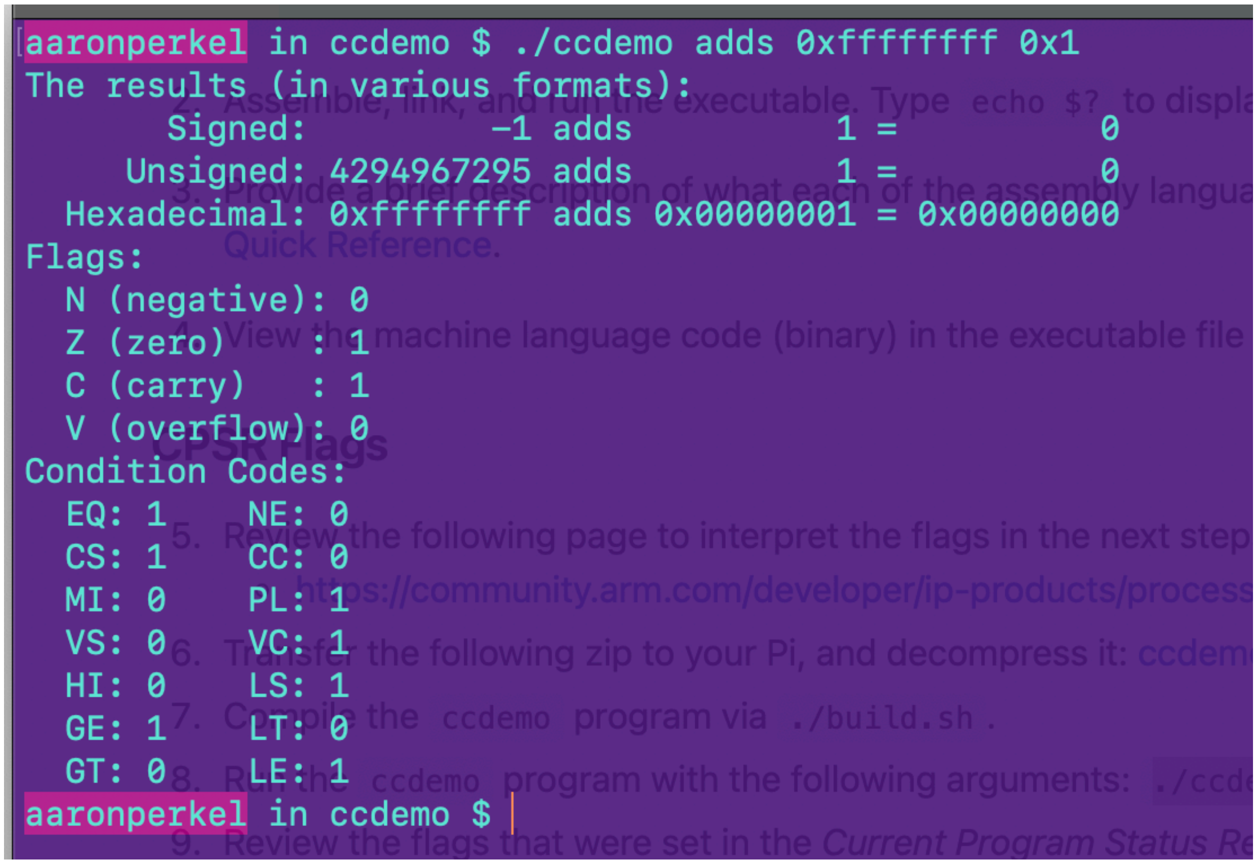
2. loop.s output

3. This loop adds all of the numbers from 1 to 22, inclusive. MOV moves data into a register. CMP compares two values, BGT preforms and action if val1 is greater than val2, ADD is addition, B moves to an address, SWI is software interrupt.



```
0000014a: 00000000 00000000 00010000 00000000 00000001 00000000  ......
00000150: 00110110 00000000 00000000 00000000 01111100 00000000  6...|.
00000156: 00000010 00000000 00000000 00000000 00000000 00000000  ......
0000015c: 00010000 00000000 00000001 00000000 00111110 00000000  ....>.
00000162: 00000000 00000000 01111100 00000000 00000010 00000000  ..|...
00000168: 00000000 00000000 00000000 00000000 00010000 00000000  ......
0000016e: 00000001 00000000 01000101 00000000 00000000 00000000  ..E...
00000174: 01111100 00000000 00000010 00000000 00000000 00000000  |.....
0000017a: 00000000 00000000 00010000 00000000 00000001 00000000  ......
00000180: 00000000 01101100 01101111 01101111 01110000 00101110  .loop.
00000186: 01101111 00000000 00100100 01100001 00000000 01101100  o.$a.l
0000018c: 01101111 01101111 01110000 00000000 01011111 01011111  oop.__
00000192: 01100010 01110011 01110011 01011111 01110011 01110100  bss_st
00000198: 01100001 01110010 01110100 01011111 01011111 00000000  art__.
0000019e: 01011111 01011111 01100010 01110011 01110011 01011111  __bss_
000001a4: 01100101 01101110 01100100 01011111 01011111 00000000  end__.
000001aa: 01011111 01011111 01100010 01110011 01110011 01011111  __bss_
000001b0: 01110011 01110100 01100001 01110010 01110100 00000000  start.
000001b6: 01011111 01011111 01100101 01101110 01100100 01011111  __end_
000001bc: 01011111 00000000 01011111 01100101 01100100 01100001  _._eda
000001c2: 01110100 01100001 00000000 01011111 01100101 01101110  ta._en
000001c8: 01100100 00000000 00000000 00101110 01110011 01111001  d...sy
000001ce: 01101101 01110100 01100001 01100010 00000000 00101110  mtab..
000001d4: 01110011 01110100 01110010 01110100 01100001 01100010  strtab
000001da: 00000000 00101110 01110011 01101000 01110011 01110100  ..shst
000001e0: 01110010 01110100 01100001 01100010 00000000 00101110  rtab..
000001e6: 01110100 01100101 01111000 01110100 00000000 00101110  text..
```

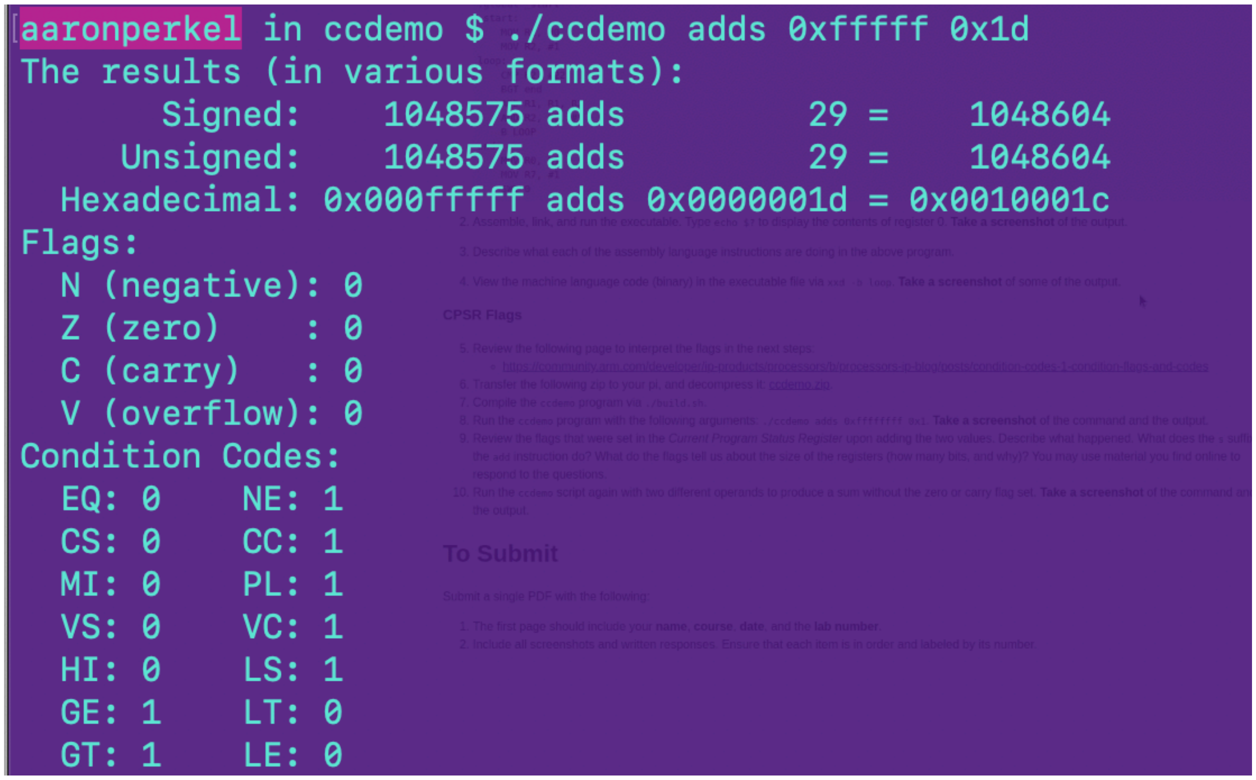
4. some output from xxd -b loop



```
aaronperkel in ccdemo $ ./ccdemo adds 0xffffffff 0x1
The results (in various formats):
       Signed:         -1 adds          1 =          0
     Unsigned: 4294967295 adds          1 =          0
  Hexadecimal: 0xffffffff adds 0x00000001 = 0x00000000
Flags:
  N (negative): 0
  Z (zero)    : 1
  C (carry)   : 1
  V (overflow): 0
Condition Codes:
  EQ: 1    NE: 0
  CS: 1    CC: 0
  MI: 0    PL: 1
  VS: 0    VC: 1
  HI: 0    LS: 1
  GE: 1    LT: 0
  GT: 0    LE: 1
aaronperkel in ccdemo $
```

8. ccdemo first run

9. Z because the result is 0, C because the addition caused it to “wrap around” back to 0. “ADDS” (as opposed to “ADD”) preforms addition, but also updates the flags after.



```
aaronperkel in ccdemo $ ./ccdemo adds 0xfffff 0x1d
The results (in various formats):
       Signed:    1048575 adds         29 =    1048604
     Unsigned:    1048575 adds         29 =    1048604
  Hexadecimal: 0x000fffff adds 0x0000001d = 0x0010001c
Flags:
  N (negative): 0
  Z (zero)    : 0
  C (carry)   : 0
  V (overflow): 0
Condition Codes:
  EQ: 0    NE: 1
  CS: 0    CC: 1
  MI: 0    PL: 1
  VS: 0    VC: 1
  HI: 0    LS: 1
  GE: 1    LT: 0
  GT: 1    LE: 0
```

10. ccdemo second run