2.8

Translate 0xabcdef12 into decimal.

$(ABCDEF12)_{16}$

$= (10 \times 16^7) + (11 \times 16^6) + (12 \times 16^5) + (13 \times 16^4) + (14 \times 16^3) + (15 \times 16^2) + (1 \times 16^1) + (2 \times 16^0)$

= 2882400018

2882400018

2.16

Provide the type, assembly language instruction, and binary representation of instruction described by the following MIPS fields:

Op = 0, rs = 3, rt = 2, rd = 3, shamt = 0, funct = 34

Type: R-type

Assembly: sub $v1, $v1, $v0

Binary: 000000 00011 00010 00011 00000 100010

2.18(all sub-parts)

2.18.1

OP will change to **8 bits** to contain four times as many instructions (shift left logical twice)

rs, rt, rd will change to **7 bits** to contain 128 registers (1111111)

shamt and funct **will not change** their bit size

2.18.2

OP will change to **8 bits** to contain four times as many instructions

rs, rt, will change to **7 bits** to contain 128 registers (1111111)

constant or address **will not change** their bit size

2.18.3

Decrease the size:

More registers:

Program may decrease in size because more registers will let you handle more data in the process without going out to memory. In this case, we will have fewer load/store instructions comparing with less registers.

More instruction sets:

In this case, we might have a faster way to solve a procedure. For example, in the past we need 5 instructions to do it, but now we only need 1 instruction to achieve what we want. Even though the bits increased to 8, but overall size of a file is decreased because there are obviously less instructions used.

Increase the size:

More registers/instruction sets:

Longer instruction and registers encoding will lead to a longer program that can do the same operations. If the total number of instructions used are the same, a longer instruction (2 more bits each) and registers (2 more bits each) will surely have a bigger size of a file.

2.25

2.25.1

I-type

2.25.2

```
blt $t2, $zero, done
addi $t2, $t2, -1
j loop
done:
```