CS Basics - Homework Assembler

E. Benoist and C. Grothoff

October 5, 2018

The goal of this work is to provide two programs in assembler for Linux 64-bit, for encoding and decoding binary files into Base32.

Base32 encoding is used to transfer binary data in text files. This is for instance used in cryptography to write keys or signature in emails or inside pdf documents.

Base32 principles

In base32 encoding, one can not print one byte in one character. One character can have only 32 possibilities (26 letters, 6 digits 2–7). So, we divide a file in groups of 5 bits ($2^5 = 32$). Each group is then transformed into a character that is given as output.

Example: Suppose, we have a file containing the following bytes (noted in hexadecimal):

```
7C AA 78
```

It can also be seen as binary information (just transform hex in binary)

```
0111 1100 1010 1010 0111 1000
```

The transformation in base64 requires that we group the bits in groups of 5 bits.

```
01111 10010 10101 00111 1000[0]
```

We can see each of these groups as a value (which number it represents)

```
15 18 21 7 16
```

Then we use a huge table:

0	A	10	K	20	U		
1	B	11	L	21	V		
2	C	12	M	22	W		
3	D	13	N	23	X		
4	E	14	O	24	Y	30	6
5	F	15	P	25	Z	31	7
6	G	16	Q	26	2		
7	H	17	R	27	3		
8	I	18	S	28	4		
9	J	19	T	29	5		

We search in the table the coding for each of the values we have

```
PSVHQ===
```

Problem with the length We code only groups of 5 bits whereas we have bytes (8 bits) in our files. In the normal case, we treat 5 bytes in a time (i.e. $8 \times 5 = 40$ bits), they are transformed into 8 characters, each one representing 5 bits ($5 \times 8 = 40$). We may have a problem with the end of the file, since the number of bytes may not be divisible by 5. In this case:

- We add bits 0 in the end, such that we arrive at a multiple of 5 bits for the total.
- We add a number of "=" (pad) signs depending on the input size (RFC 4648):
  1. The final quantum of encoding input is an integral multiple of 40 bits; here, the final unit of encoded output will be an integral multiple of 8 characters with no "=" padding.
  2. The final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by six "=" padding characters.
  3. The final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output will be four characters followed by four "=" padding characters.
  4. The final quantum of encoding input is exactly 24 bits; here, the final unit of encoded output will be five characters followed by three "=" padding characters.
  5. The final quantum of encoding input is exactly 32 bits; here, the final unit of encoded output will be seven characters followed by one "=" padding character.

more information can be found in RFC 4648: `https://tools.ietf.org/html/rfc4648`

Exercise

You have to write an assembly language program that encodes binary into base32, and another program deconding a base32 file into a binary.

The input will be read from standard input (like in all our exercises) and output is written in the standard output (also like in all our exercises)

This exercise is to be done in a group of two students. If a class has an odd number of students, one student will work alone.

The program must be written in Assembly language, the only exception may be the use of the two C functions `printf()` and `scanf()`.

You must deliver the following files:

- `base32enc.asm` your program to encode into Base32
- `base32dec.asm` your program to decode from Base32

You should TEST your code with the provided `test.sh` (and if possible extend the test logic with further tests, especially for the decoder). Your code must link and run with the provided script. If it does not, you will get ZERO points. You must `apt-get install nasm base32` to run the provided test script.

Deadline: 22nd of December 2018, the students must send to the professor a zip file containing a directory with the two programs.