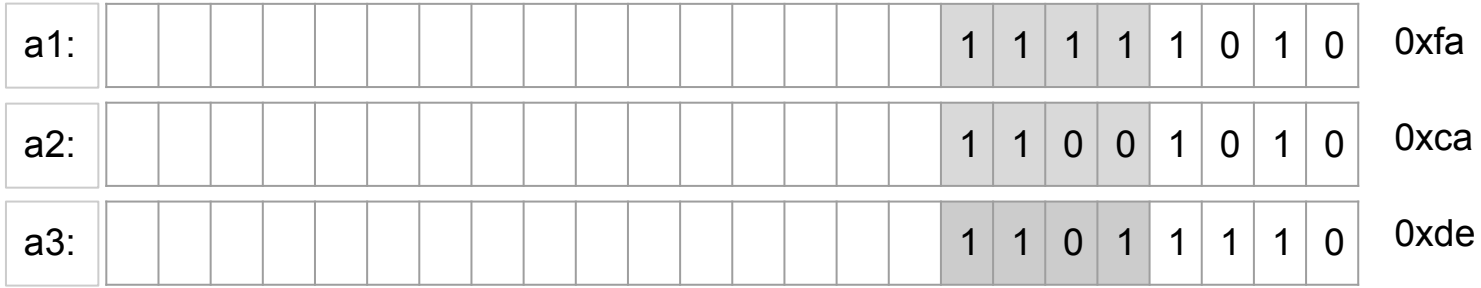
Base64: a binary string is encoded as an ASCII string

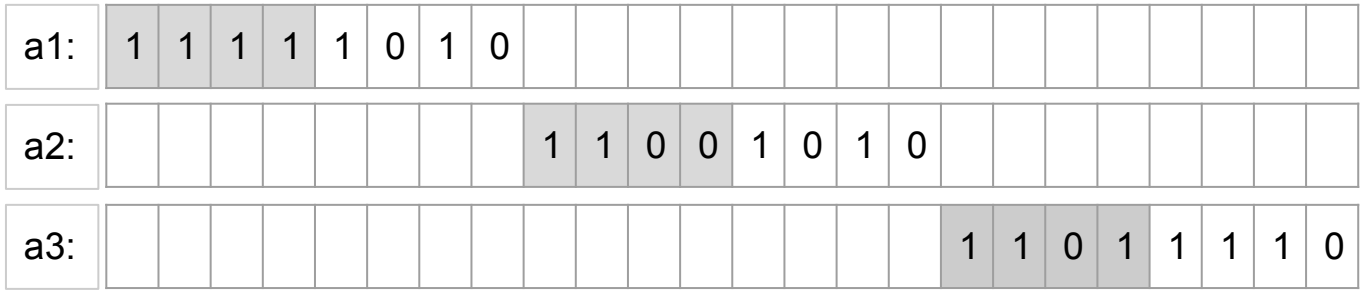
- Basic Algorithm:
  - For every three bytes (24 bits) ? lcm(6,8) =
    - Load and Merge the bytes together
    - Chop and Slide into 4 6-bit chunks
    - Map each 6-bit chunks into a 8-bit ASCII value
    - Store each new the original three bytes with four new bytes
  - Add appropriate padding for remaining bytes
- Mapping ensures the result 8 bits are always printable ASCII characters
- Operations at the assemble level:
  - byte manipulations
  - shifting and masks
- Working at the byte level exposes Endianness

Load and Merge (shift and meld)

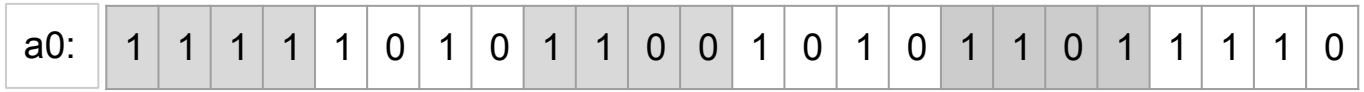
- load:



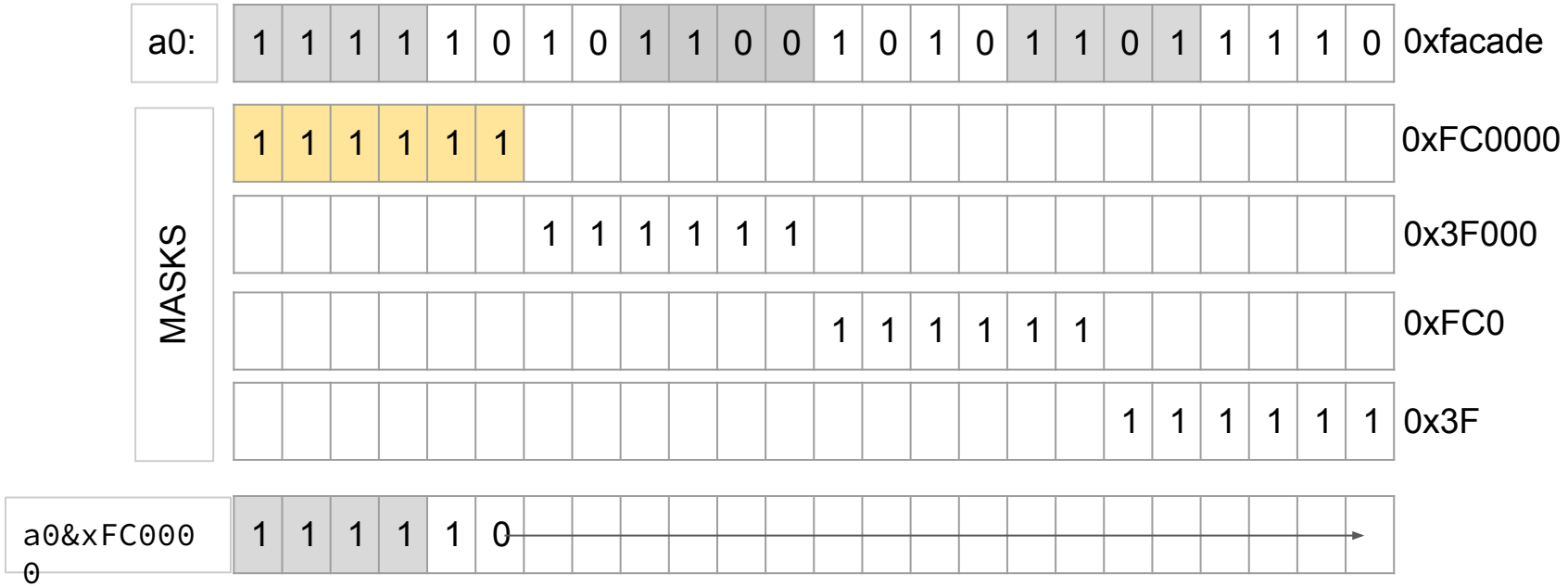
- shift:



- meld:



Chop and Slide:




```
s1= (a0 & xFC0000) >> 18= 11 1110= 0x3E
```

```
s2= (a0 & x03F000) >> 12= 10 1100= 0x2C
```

```
s3= (a0 & xFC0) >> 6 == 10 1011= 0x2B
```

```
s4= (a0 & x03F) >> 0 == 01 1110= 0x1E
```

Map:

Why?
65 is the ASCII code for 'A'

- For simplicity, I have just added 'A' (65) to each chunk
- Real mapping

```
switch ( chunk ) {
  0..25  : chunk += 0 + 'A' ; // A= 65 // A - Z
           break;
  26..51 : chunk += -26 + 'a'; // a - z
           break;
  52..61 : chunk += -52 + '0'; // 0 - 9
           break;
  62     : chunk = '+';
           break;
  63     : chunk = '/';
}
```

Store

- Well, that is straightforward and simple enough

```
la $t0, output
sb $s1, 0($t0)
sb $s2, 1($t0)
sb $s3, 2($t0)
sb $s4, 3($t0)
```