Roadmap

C:

```
car *c = malloc(sizeof(car));
c->miles = 100;
c->gals = 17;
float mpg = get_mpg(c);
free(c);
```

Java:

```
Car c = new Car();
c.setMiles(100);
c.setGals(17);
float mpg =
    c.getMPG();
```

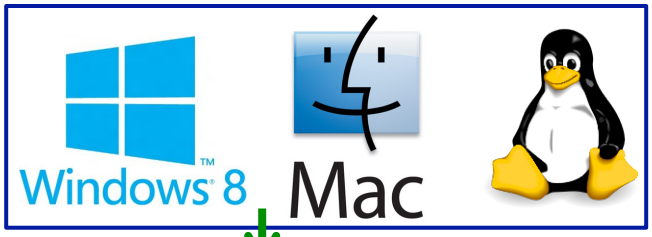
Assembly language:

```
get_mpg:
    pushq   %rbp
    movq    %rsp, %rbp
    ...
    popq    %rbp
    ret
```

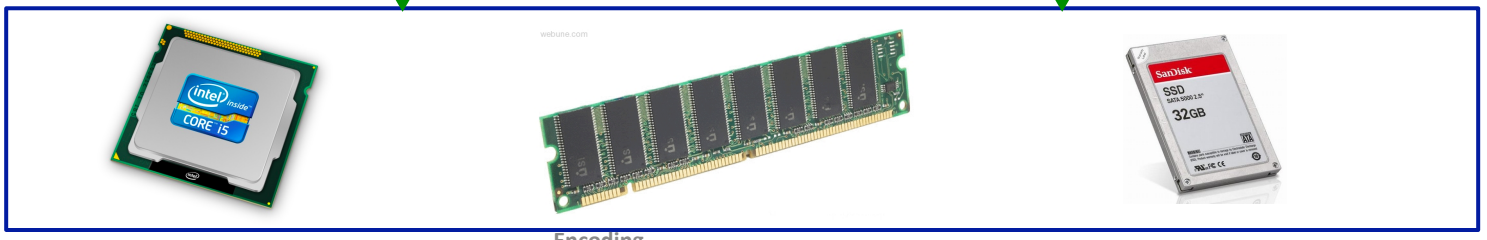
Machine code:

```
0111010000011000
10001101000010000000010
1000100111000010
11000001111110100011111
```

OS:



Computer system:



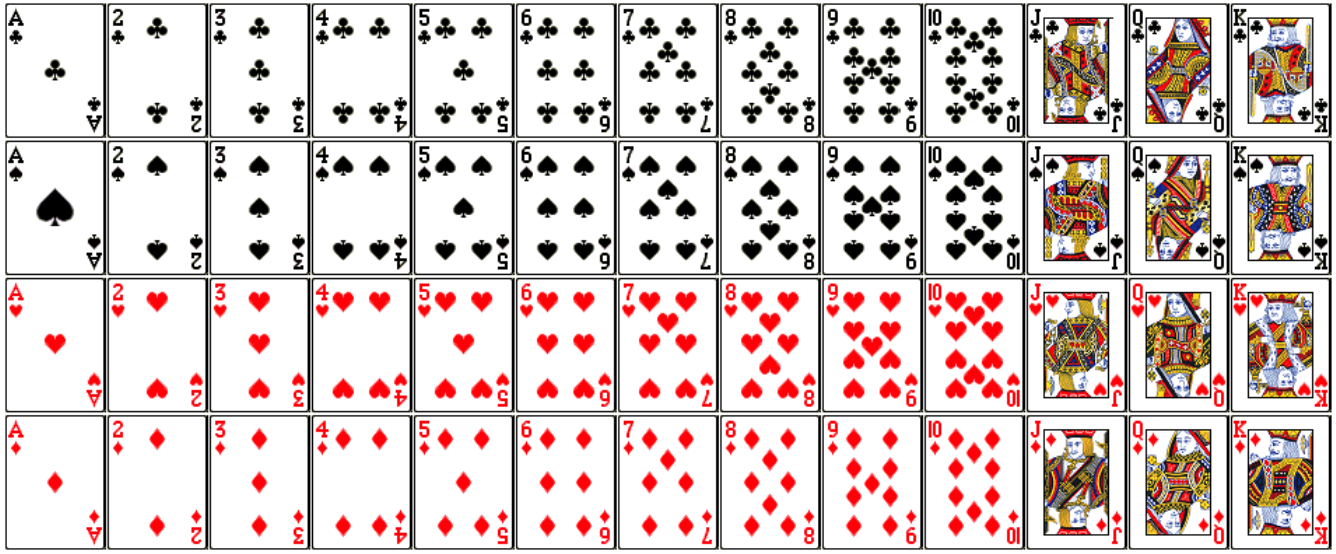
Section 2: Integer & Floating Point Numbers

- Representation of integers: unsigned and signed
- Unsigned and signed integers in C
- Arithmetic and shifting
- Sign extension

- Background: fractional binary numbers
- IEEE floating-point standard
- Floating-point operations and rounding
- Floating-point in C

But before we get to integers….

- **How about encoding a standard deck of playing cards?**
- **52 cards in 4 suits**
  - How do we encode suits, face cards?
- **What operations do we want to make easy to implement?**
  - Which is the higher value card?
  - Are they the same suit?



Two possible representations

- **52 cards – 52 bits with bit corresponding to card set to 1**

  - "One-hot" encoding
  - Drawbacks:
    - Hard to compare values and suits
    - Large number of bits required

- **4 bits for suit, 13 bits for card value – 17 bits with two set to 1**

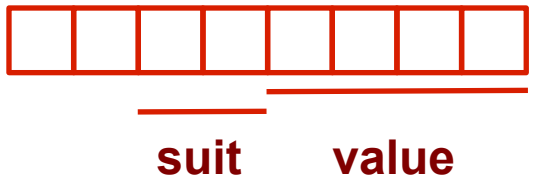
  - "Two-hot" encoding
  - Easier to compare suits and values
    - Still an excessive number of bits

Two better representations

- **Binary encoding of all 52 cards – only 6 bits needed**

  - Fits in one byte
  - Smaller than one-hot or two-hot encoding, but how can we make value and suit comparisons easier?

- **Binary encoding of suit (2 bits) and value (4 bits) separately**

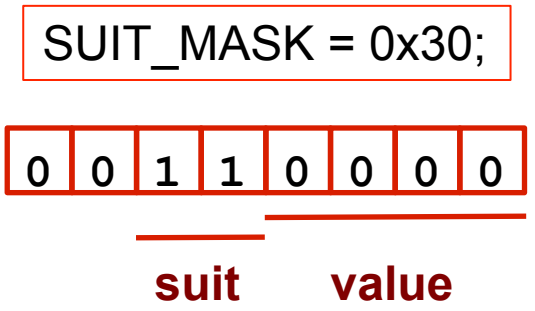


  - Also fits in one byte, and easy to do comparisons

Some basic operations

- **Checking if two cards have the same suit:**

```
#define SUIT_MASK  0x30
char array[5];       // represents a 5 card hand
char card1, card2;   // two cards to compare
card1 = array[0];
card2 = array[1];
...
if sameSuitP(card1, card2) {
...
```




```
bool sameSuitP(char card1, char card2) {
  return (! (card1 & SUIT_MASK) ^ (card2 & SUIT_MASK));
   //return (card1 & SUIT_MASK) == (card2 & SUIT_MASK);
}
```

Some basic operations

- **Comparing the values of two cards:**

```
#define SUIT_MASK   0x30
#define VALUE_MASK 0x0F
char array[5];        // represents a 5 card hand
char card1, card2;    // two cards to compare
card1 = array[0];
card2 = array[1];
...
if greaterValue(card1, card2) {
...
bool greaterValue(char card1, char card2) {
  return ((unsigned int)(card1 & VALUE_MASK) >
          (unsigned int)(card2 & VALUE_MASK));
}
```