C - Bit Fields, xv6 Setup, gdb

CS238P: Principles of operating systems - Fall'18

Aftab Hussain
(Adapted from Vikram Narayanan's CS143A Fall '17 slides)

October 26, 2018

University of California, Irvine

Bits

A data structure to hold bits

Bit fields[1]

```
// Gate descriptors for interrupts and traps
struct gatedesc {
  uint off_15_0 : 16; // low 16 bits of offset in segment
  uint cs : 16; // code segment selector
  uint args : 5; // # args, 0 for interrupt/trap gates
  uint rsv1 : 3; // reserved(should be zero I guess)
  uint type : 4; // type(STS_{TG,IG32,TG32})
  uint s : 1; // must be 0 (system)
  uint dpl : 2; // descriptor(meaning new) privilege level
  uint p : 1; // Present
  uint off_31_16 : 16; // high bits of offset in segment
};

struct gatedesc d;
d.s = 0; d.args = 0;
```

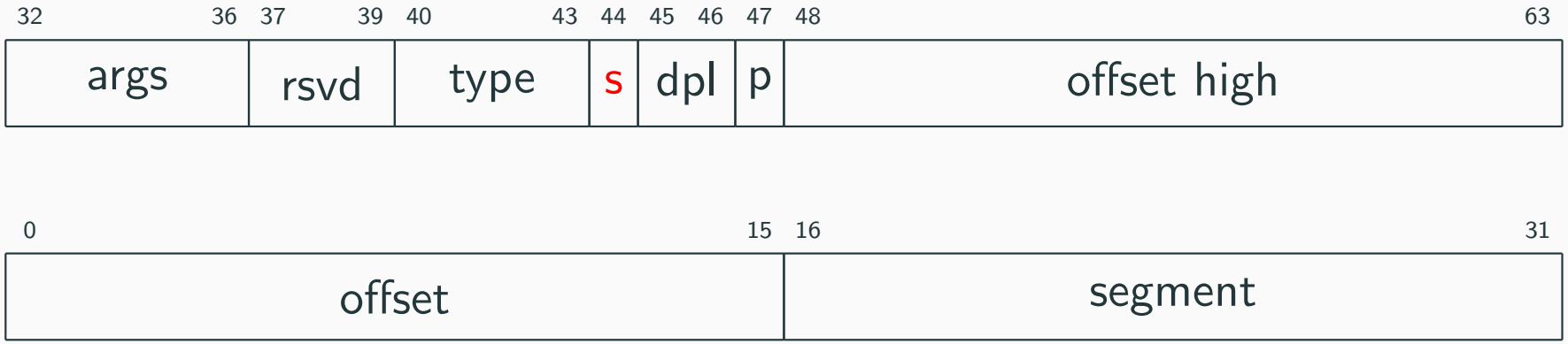
[1] sheet 09 xv6-rev9.pdf

Bit fields

"In C and C++, native implementation-defined bit fields can be created using unsigned int, signed int, or (in C99:) _Bool."[2]

[2] https://en.wikipedia.org/wiki/Bit_field

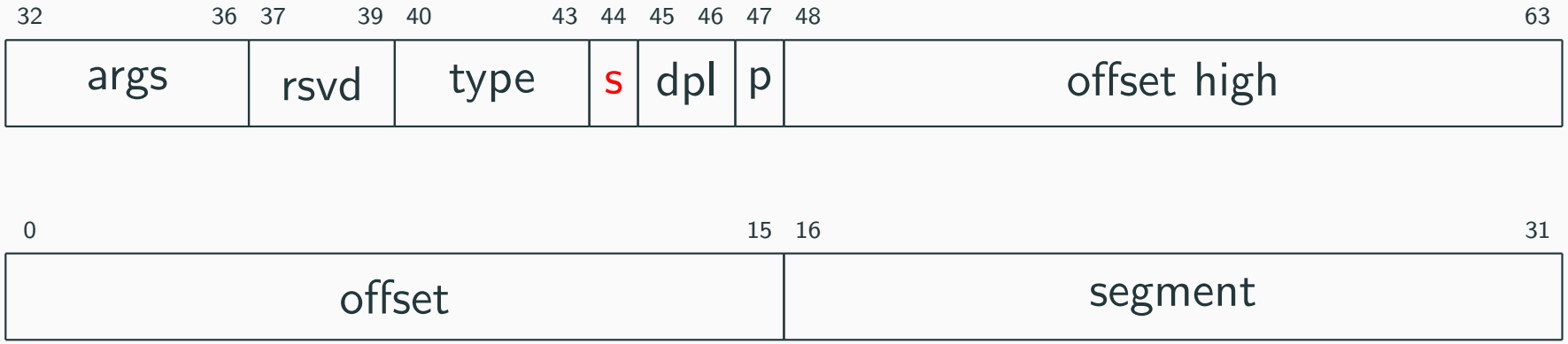
Access low-level data



- Set bit 44 (s) to 1, without changing anything else.

```
/* on a 64-bit data type */
data = data | (1 << 44); //the same as under
data |= (1 << 44);
```

Access low-level data



- Set bit 44 (s) to 1, without changing anything else.

```
/* on a 64-bit data type */
data = data | (1 << 44); //the same as under
data |= (1 << 44);
```

- Clear a bit (s) - And (&) and Not (∼)

```
/* on a 64-bit data type */
data = data & ~(1 << 44);
data &= ~(1 << 44);
```

Moving on to xv6 setup and gdb demo.