C programming

CS143A: Principles of operating systems - Fall'17

UC Irvine, California

Hw1(xv6 shell)

- if...else

```
pid = fork();
if(pid == -1)
  perror("fork:");
```

Hw1(xv6 shell)

- if...else

```
pid = fork();
if(pid == -1)
  perror("fork:");
```

- switch...case

```
switch(cmd->type){
case '>': ...; break;
default: ...; break;
}
```

Hw1(xv6 shell)

- `if...else`

```
pid = fork();
if(pid == -1)
  perror("fork:");
```

- `switch...case`

```
switch(cmd->type){
case '>': ...; break;
default: ...; break;
}
```

- Functions
  - Process creation (`fork`, `exec`)
  - File I/O (`open`, `close`, `read`, `write`)

```
fd = open(rcmd->file, rcmd->mode);
```

Hw1(xv6 shell)

- `if...else`

```
pid = fork();
if(pid == -1)
  perror("fork:");
```

- `switch...case`

```
switch(cmd->type){
case '>': ...; break;
default: ...; break;
}
```

- Functions
  - Process creation (`fork`, `exec`)
  - File I/O (`open`, `close`, `read`, `write`)

    ```
    fd = open(rcmd->file, rcmd->mode);
    ```

- Typecasting (next slide)

Hw1(xv6 shell)

- `if...else`

```
pid = fork();
if(pid == -1)
  perror("fork:");
```

- `switch...case`

```
switch(cmd->type){
case '>': ...; break;
default: ...; break;
}
```

- Functions
  - Process creation (`fork`, `exec`)
  - File I/O (`open`, `close`, `read`, `write`)

```
fd = open(rcmd->file, rcmd->mode);
```

- Typecasting (next slide)
- Command line arguments (`argv`)

Typecasting

- Change the type of the object for a single operation

```
var = (dest_type) source;
```

Typecasting

- Change the type of the object for a single operation

```
var = (dest_type) source;
```

- Pass generic objects

```
struct cmd { int type; };
struct execcmd {
  int type;
  char *argv[MAXARGS];
};
void runcmd(struct cmd *cmd) {
    ...
    ecmd = (struct execcmd*)cmd;
}
struct cmd* execcmd(void) {
  struct execcmd *cmd;
  ...
  return (struct cmd*)cmd;
}
```

Typecasting

- Change the type of the object for a single operation

```
var = (dest_type) source;
```

- Pass generic objects

```
struct cmd { int type; };
struct execcmd {
  int type;
  char *argv[MAXARGS];
};
void runcmd(struct cmd *cmd) {
    ...
    ecmd = (struct execcmd*)cmd;
}
struct cmd* execcmd(void) {
  struct execcmd *cmd;
  ...
  return (struct cmd*)cmd;
}
```

- Beware of strings! (demo: str.c)

Arrays

- Collection of objects of the same data type

Arrays

- Collection of objects of the same data type
- Accessed by index (`0 ... size - 1`)

Arrays

- Collection of objects of the same data type
- Accessed by index (`0 ... size - 1`)
- String is an array of characters (demo: string.c)

Arrays

- Collection of objects of the same data type
- Accessed by index (`0 ... size - 1`)
- String is an array of characters (demo: string.c)
- No reference operator

```
printf("Address of a \%p | \%p\n", a, &a);
>> Address of a 0x7aff07024060 | 0x7aff07024060
```

Array Intialization

Designated Initializers[1]

```
#define CAPSLOCK (1<<3)
#define NUMLOCK (1<<4)
#define SCROLLLOCK (1<<5)
static uchar togglecode[256] = {
[0x3A] CAPSLOCK,
[0x45] NUMLOCK,
[0x46] SCROLLLOCK
};
/* equivalent to */
togglecode[0x3A] = CAPSLOCK;
togglecode[0x45] = NUMLOCK;
togglecode[0x46] = SCROLLLOCK;
```

Initialize the array elements 0x3A, 0x45, 0x46 only [2]

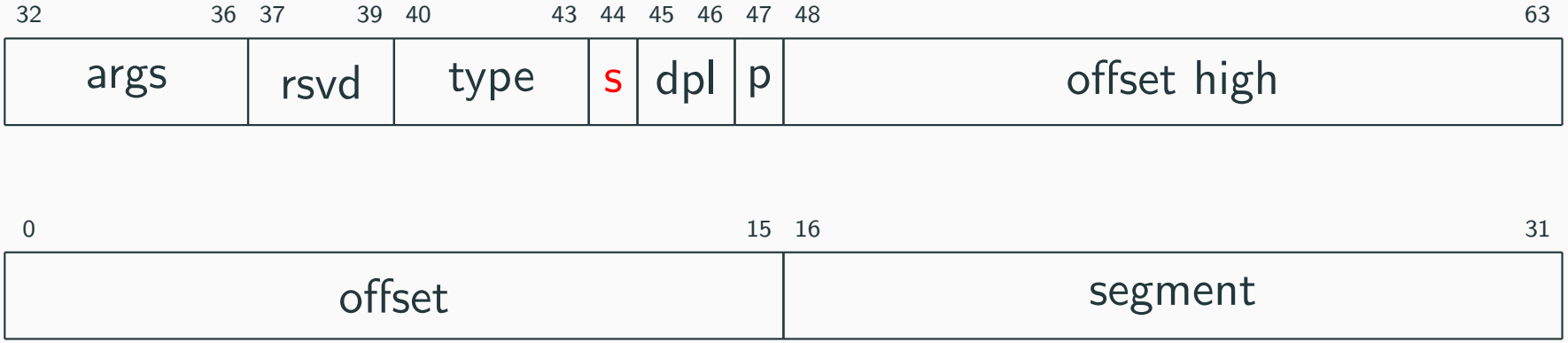
[1] http://gcc.gnu.org/onlinedocs/gcc-4.0.4/gcc/Designated-Inits.html
[2] sheet 77, xv6-rev9.pdf

Bit fields[3]

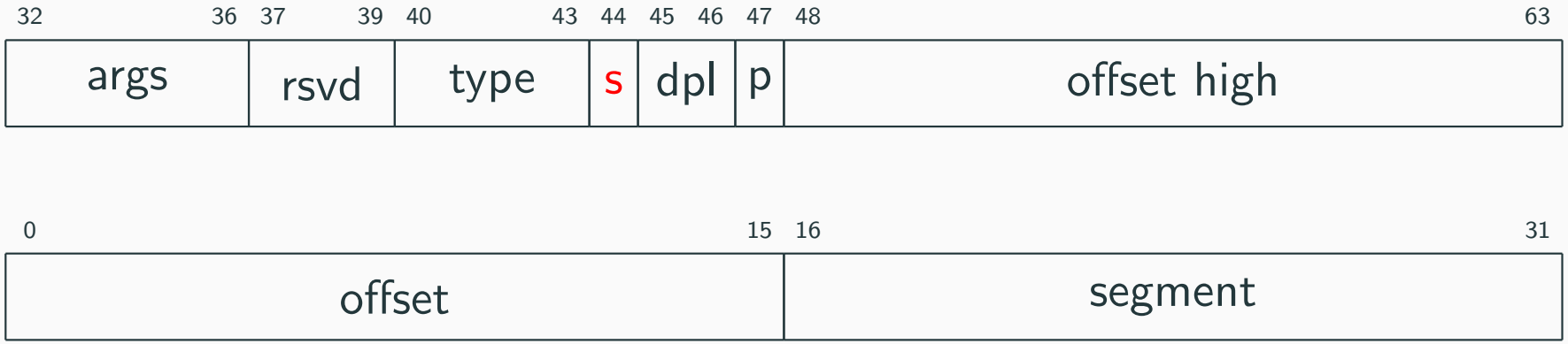
```
// Gate descriptors for interrupts and traps
struct gatedesc {
  uint off_15_0 : 16; // low 16 bits of offset in segment
  uint cs : 16; // code segment selector
  uint args : 5; // # args, 0 for interrupt/trap gates
  uint rsv1 : 3; // reserved(should be zero I guess)
  uint type : 4; // type(STS_{TG,IG32,TG32})
  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;
```

[3] sheet 09 xv6-rev9.pdf

Access low-level data



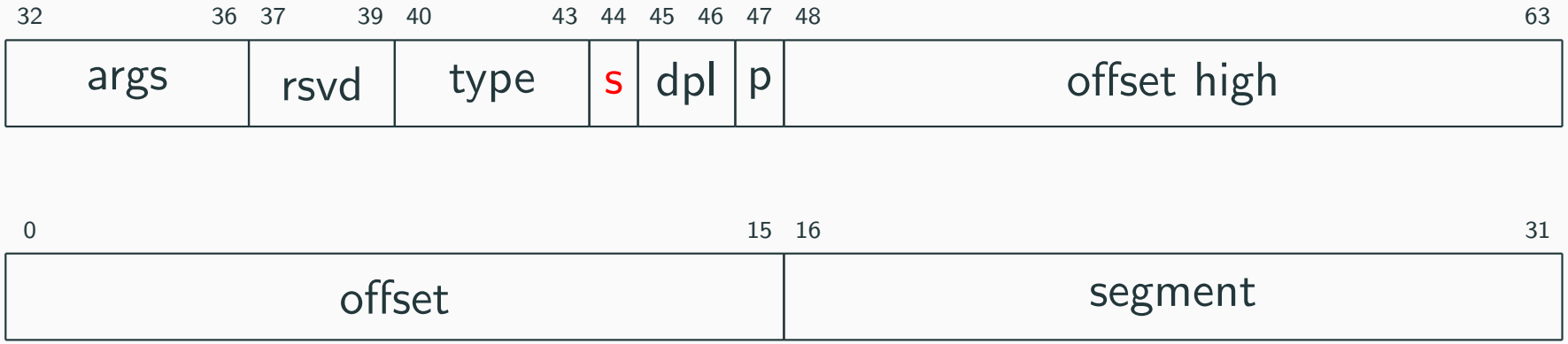
Access low-level data



- Set bit 44 (s) - Or (|) it

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

Access low-level data



- Set bit 44 (s) - Or (|) it

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

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

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

Dynamic registration

- Declare a struct to hold function pointers [4]

```
#define NDEV  10
#define CONSOLE 1
struct devsw {
  int (*read)(struct inode*, char*, int);
  int (*write)(struct inode*, char*, int);
};
struct devsw devsw[NDEV]; /* global data structure */
```

[4] sheet 40 xv6-rev9.pdf
[5] sheet 82 xv6-rev9.pdf

Dynamic registration

- Declare a struct to hold function pointers [4]

```
#define NDEV  10
#define CONSOLE 1
struct devsw {
  int (*read)(struct inode*, char*, int);
  int (*write)(struct inode*, char*, int);
};
struct devsw devsw[NDEV]; /* global data structure */
```

- Register function pointer [5]

```
int consolewrite(struct inode *ip, char *buf, int n);
int consoleread(struct inode *ip, char *dst, int n);
devsw[CONSOLE].write = consolewrite;
devsw[CONSOLE].read = consoleread;
```

[4] sheet 40 xv6-rev9.pdf
[5] sheet 82 xv6-rev9.pdf

Pointers & buffer management

- Access raw memory

```
#define KERNBASE 0x80000000
#define P2V(a) (((void *) (a)) + KERNBASE)
uchar *code;
code = P2V(0x7000);
```

Pointers & buffer management

- Access raw memory

```
#define KERNBASE 0x80000000
#define P2V(a) (((void *) (a)) + KERNBASE)
uchar *code;
code = P2V(0x7000);
```

- kalloc, memset, kfree

```
mem = kalloc(); /* allocate a page */
memset(mem, 0, PGSIZE); /* memset */
kfree(mem); /* free it when done */
```

Pointers & buffer management

- Access raw memory

```
#define KERNBASE 0x80000000
#define P2V(a) (((void *) (a)) + KERNBASE)
uchar *code;
code = P2V(0x7000);
```

- kalloc, memset, kfree

```
mem = kalloc(); /* allocate a page */
memset(mem, 0, PGSIZE); /* memset */
kfree(mem); /* free it when done */
```

- memcpy, memmove

```
/* move start to code */
memmove(code, _binary_entryother_start,
      (uint)_binary_entryother_size);
```

Questions?