Sunbeam Institute of Information Technology
Pune and Karad

Module - Embedded C Programming

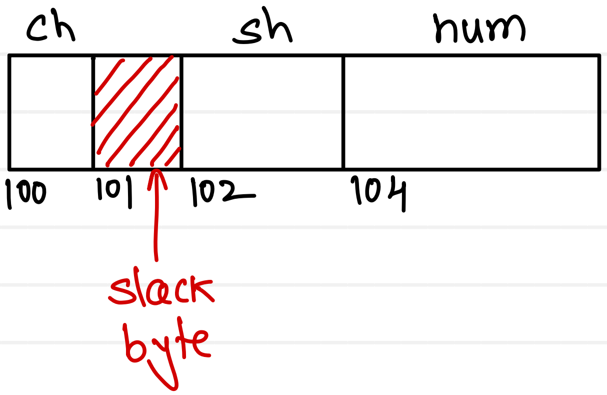
Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

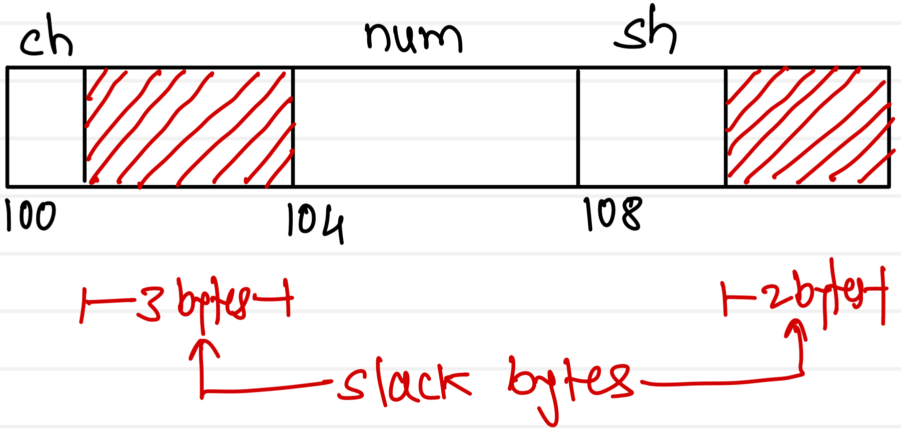
Structure padding

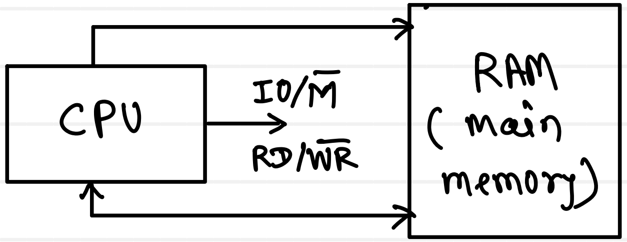
- compiler adds extra bytes in structure variables to make their sizes multiple of largest data member.
- for word aligned access also, variable should be stored on address which is multiple of its size.

```
struct test {
        char ch;
        short sh;
        int num;
};

  struct test  t ;
```



```
struct test {
        char ch;
        int num;
        short sh;
};

  struct test  t ;
```





Word unaligned access:

CPU can access data of any size from any address.

word aligned access:

CPU can access data of size from address which is multiple of size.

e.g. for int, address will be multiple of 4

for short, address will be multiple of 2

```
struct test {
    int a;
    char b;
    short c;
    int d;
    short e;
    char f;
    int g;
}t;
```

100	a	a	a	a
104	b	c	c	d
108	d	d	d	e
112	e	f	g	g
116	g	g		
120				
124				

t.d

1. read word from 104
2. read word from 108
3. find value of d

100	a	a	a	a
104	b	▨	c	c
108	d	d	d	d
112	e	e	f	▨
116	g	g	g	g
120				
124				

t.d

1. read word from 108

Array of structure

```
struct player {
        char name[20];
        int age;
        ptype_t type;
        int matches;
} arr[3];
```

arr[0] - 1st player {"abc", 28, 0, 122}

arr[1] - 2nd player

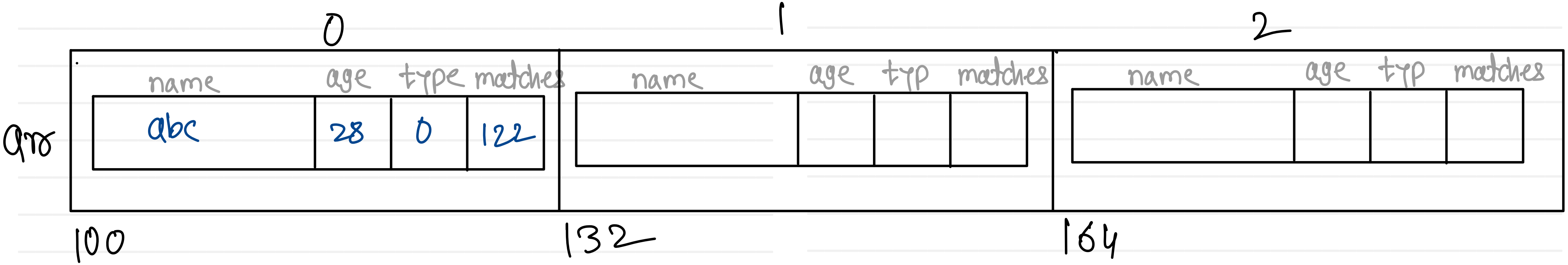
arr[2] - 3rd player

arr[0].name = "abc"

arr[0].age = 28

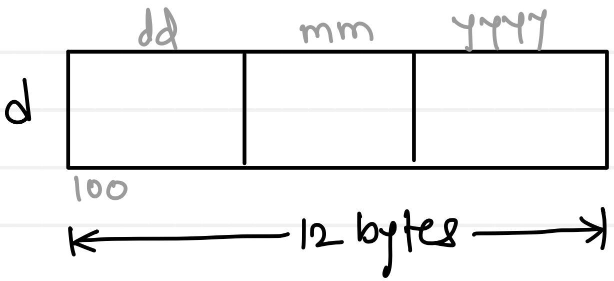
arr[0].type = 0

arr[0].matches = 122

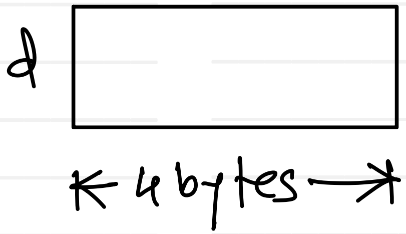


Bit fields

```
struct date {
    unsigned int dd;
    unsigned int mm;
    unsigned int yyyy;
};
struct date d;
```



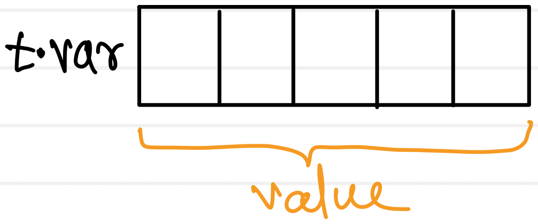
```
struct date {
    unsigned int dd : 5 ;
    unsigned int mm : 4 ;
    unsigned int yyyy : 16;
};
                  25 bits
struct date d;
```



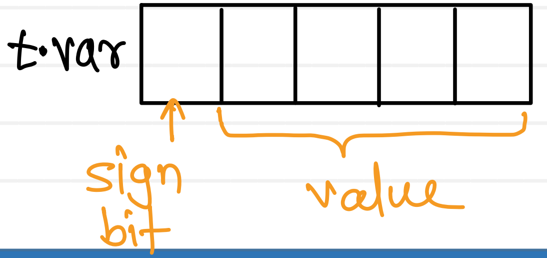
```
d.dd = 14
d.mm = 3
d.yyyy=2025
```

```
&d.dd
&d.mm     } X
&d.yyyy
```

```
struct test {
    unsigned int var :5;
}t;
```



```
struct test {
    int var :5;
}t;
```



Union

- union is collection of similar or dissimilar type of data.

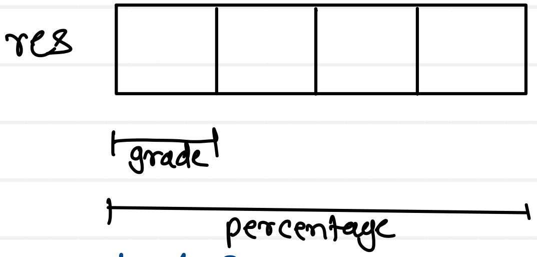
  $\text{sizeof(struct)} = \sum \text{sizeof(members)}$

  sizeof(union) = sizeof(largest member)

- same space is shared by all the members of union.
- At a time value of only one member will be valid.
- updating one member will affect others

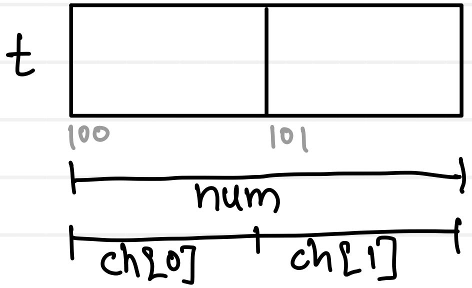
```
union result {
        char grade;
        float percentage;
}res;
```

size = 4 bytes



```
union test {
        short num;
        char ch[2];
}t;
```

size=2 bytes



SUNBEAM

Thank you!!!

Devendra Dhande

devendra.dhande@sunbeaminfo.com