PROBLEM SOLVING USING C

UNIT-5

Vibha Masti

Feedback/corrections: vibha@pesu.pes.edu

Union

- not all members required
- memory saved
- used for sharing of memory
- similar to structure

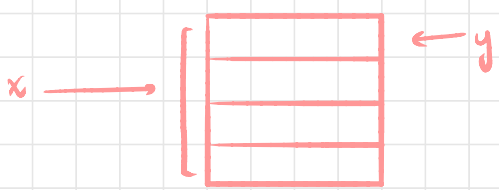
```
union xyz {
    int x;
    char y;
};
```

int — 4
char — 1
total — 5

in union, instead of 5 bytes, 4 bytes are allocated (max. datatype size)

`union xyz xy;` → memory

for any variable of type union xyz

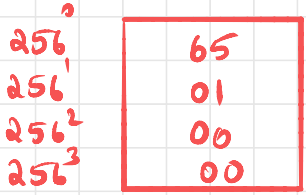


sharing always from first location

- under the assumption that not all members will be used
- memory shared

Assigning values

`xy.x = 321;`



← y gets stored here (shared)

each byte : _ _ _ _ _ _ _ _ (8-bit) (0-255)

321 = 00000000 00000000 00000001 01000001

0 0 1 65

(32-bit)

```
printf("x = %d\t y = %d\n", xy.x , xy.y);
```

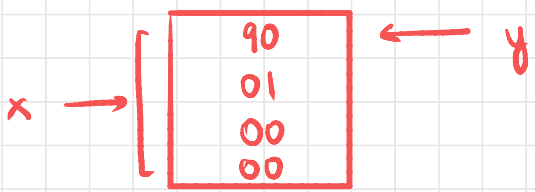
C allows

OUTPUT

321 65

Assigning other values

```
xy.y = 'Z';
```



x = 346

```
printf("x = %d\t y = %d\n", xy.x , xy.y);
```

C allows

OUTPUT

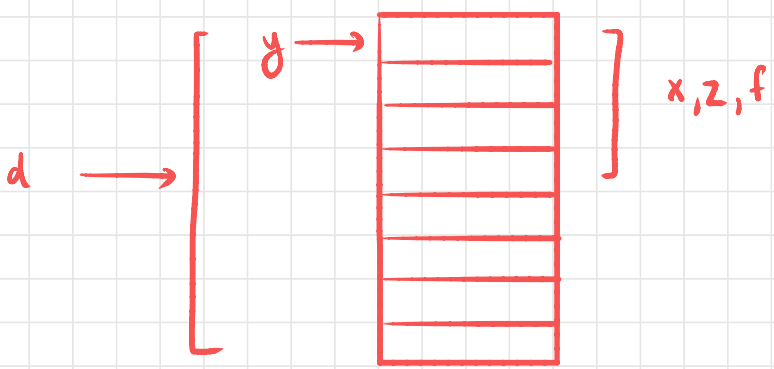
346 90

Example

```
union xyz {
    int x;
    int z;
    char y;
    double d;
    float f;
};
```

→ size of variables = 8

```
union xyz AD;
```



Note:

- even if union contains 5 chars and 1 int, the size of union variables = 4 bytes and not 5

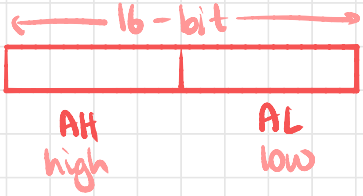
- if array / structure stored inside union, the size of the largest member (may be array or struct) is the size of union

Usage of unions

intel processors

Registers

AX
BX
CX
DX

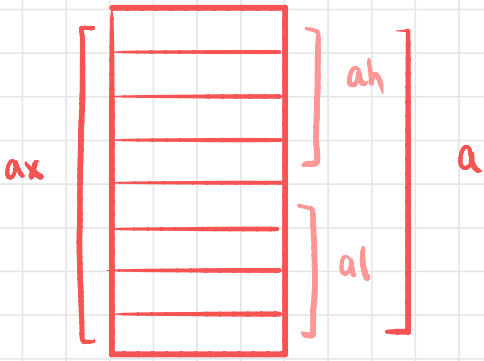


Demonstration of registers

```
union registors {

        long long ax;
        struct reg {
                int ah;
                int al;
        } a;

} A1;
```



Structure and Union

Structure: memory explicitly & uniquely allocated for each field

Union: memory allocated for largest member field and remaining fields share the allocated memory

Structures can be self-referential but unions cannot be

Bit Fields

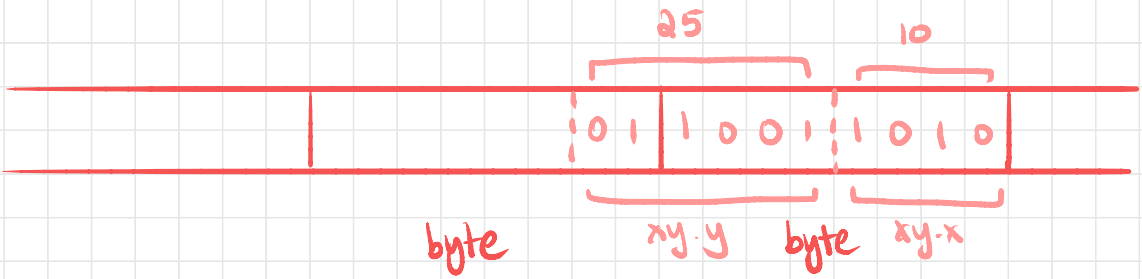
(only for integers)

```
struct xyz {
    int x;      ⟶ 4 bytes ]
    int y;      ⟶ 4 bytes ] 64 bits total
} xy;
```

Restricting size of ints to save space

```
struct xyz {
    unsigned int x: 4;   4-bits ]
    unsigned int y: 6;   6-bits ] 10 bits
} xy;
```

x: 0 - 15 (2^4)
y: 0 - 65 (2^6)



`xy.x = 20;` → error (exceeds limit)

`xy.x = 10;`
`xy.y = 25;` → allowed

(only for integers)

better in hexadecimal

note: try

- Can be used anywhere in program
- Advantage: save memory
- Must be careful

if size < 32 bits: size = 4
if size < 64 bits: size = 8