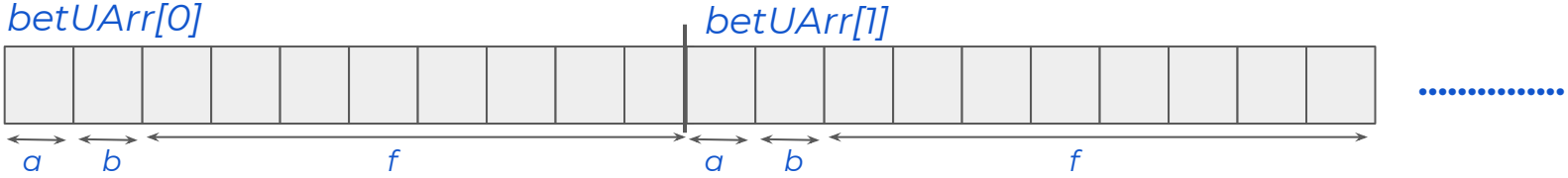
```
struct AlphaS // size of an AlphaS var is 2 + 8 = 10 bytes
                {
                        char a, b;  // contributes 2 bytes
                        double f;  // contributes 8 bytes
                };
union BetaU // size of a BetaU var is size of largest field var (alphaVar)= 10
                {
                        int i;
                        Alpha alphaVar;
                };
BetaU betUArr[25];
```

Address of **betUArr[ ].alphaVar.f** is
= Address of **betUArr[ ]** + Offsets by **a & b** in **alphaVar** struct
NOT
= Address of **betUArr[ ] +** size of **alphaVar** struct **+** Offsets by **a & b** in **alphavar** struct

See illustrations next slides..

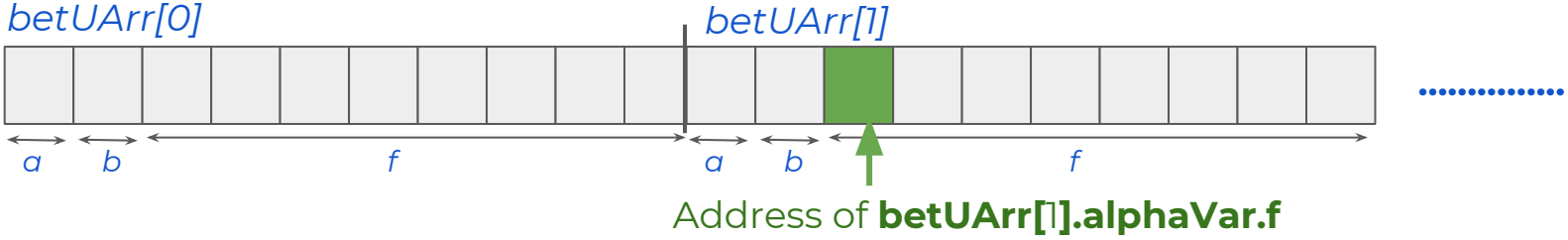
```
struct AlphaS // size of an AlphaS var is 2 + 8 = 10 bytes
                {
                        char a, b;  // contributes 2 bytes
                        double f;  // contributes 8 bytes
                };
union BetaU // size of a BetaU var is size of largest field var (alphaVar)= 10
                {
                        int i;
                        Alpha alphaVar;
                };
BetaU betUArr[25];
```

Layout of **betaUArr** array. **a**, **b**, **f** are variables of **alphaVar**



```
struct AlphaS // size of an AlphaS var is 2 + 8 = 10 bytes
                {
                        char a, b;  // contributes 2 bytes
                        double f;  // contributes 8 bytes
                };
union BetaU // size of a BetaU var is size of largest field var (alphaVar)= 10
                {
                        int i;
                        Alpha alphaVar;
                };
BetaU betUArr[25];
```

Layout of **betaUArr** array. **a**, **b**, **f** are variables of **alphaVar**



Aftab Hussain
University of California, Irvine
6th November 2018