```
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[**l**].alphaVar.f** is
= Address of **betUArr[**l**]** + Offsets by **a & b** in **alphaVar** struct
<u>NOT</u>
= Address of **betUArr[**l**] +** 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];
```
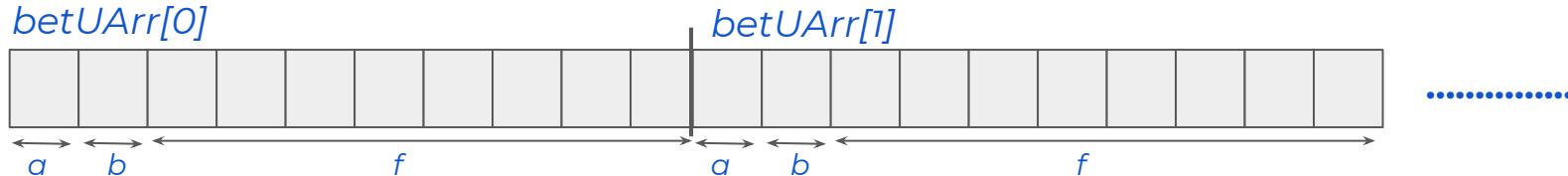
Layout of **betaUArr** array. **a**, **b**, **f** are variables of **alphaVar**

betUArr[0]   *betUArr[1]*

a   *b*   *f*   *a*   *b*   *f*

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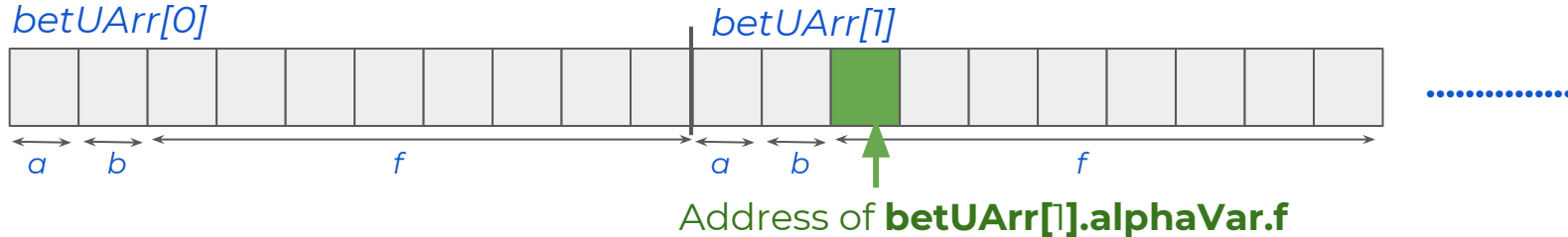
Layout of **betaUArr** array. **a**, **b**, **f** are variables of **alphaVar**

betUArr[0]   *betUArr[1]*

a   *b*   *f*   *a*   *b*   *f*

Address of **betUArr[1].alphaVar.f**

Aftab Hussain
University of California, Irvine
6th November 2017