UInt12Arrays.jl

UInt12Arrays.jl is a package for handling packed 12-bit integers. The only implemented packed format is where three bytes represents two unsigned 12-bit integers. The lower and upper four bits of the second byte are the most significant and least significant bits of the first and second integers, respectively.

```
julia> using UInt12Arrays

julia> data = UInt8[0x10, 0x32, 0x54, 0x76, 0x98, 0xBA, 0xDC, 0xFE, 0xFF];

julia> UInt12Array(data)
6-element UInt12Vector{UInt16, Vector{UInt8}}:
 0x0210
 0x0543
 0x0876
 0x0ba9
 0x0edc
 0x0fff
```

Implemented Types

```
julia> UInt12Array{UInt12}(data)
6-element UInt12Vector{UInt12, Vector{UInt8}}:
0x210
0x543
0x876
0xba9
0xedc
0xfff
```

UInt24 represents a pair of 12-bit integers

```
julia> uint24_data = reinterpret(UInt24, data)
3-element reinterpret(UInt24, ::Vector{UInt8}):
0x543210
0xba9876
0xfffedc

julia> first(uint24_data[1]), last(uint24_data[1])
(0x0210, 0x0543)
```

Accelerated unpacking

```
julia> A = UInt12Array(rand(UInt8, 1024*1024*1024*3)) # 3 GB of data
2147483648-element UInt12Vector{UInt16, Vector{UInt8}}:
 0x0d24
 0x031b
 0x056a
 0x0715
 0x0a81
 0x0408
      ⋮
 0x0295
 0x0841
 0x0c26
 0x0d9a
 0x081b
 0x0830

 julia> @time copy(A); # not accelerated
  7.973563 seconds (2 allocations: 4.000 GiB, 0.20% gc time)

julia> @time convert(Array{UInt16}, A); # SIMD-accelerated
  1.369605 seconds (4 allocations: 4.000 GiB, 14.99% gc time)
```