This example shows how to pack a struct into a list of Double Words (32 bit) keeping the **origin al order**
. This time we got less talking, and more code:

Here's our struct:

{code}

```
struct my_struct_s { %byte1: byte;
 %word1: uint(bits: 16);
 %byte2: byte;
 %byte3: byte;
 %byte4: byte;
 %data1: uint(bits:11);
 %dword1: uint(bits: 32);
 %data2: uint(bits: 5);


keep byte1 == 0x12;
 keep word1 == 0x3456;
 keep byte2 == 0x78;
 keep byte3 == 0x9a;
 keep byte4 == 0xff;
 keep data1 == 11'b10101010101;
 keep dword1 == 32'b01010111111111111111101010101010;
 keep data2 == 5'b10101;
 };
```

{/code}

Let's instantiate it and pack it:

{code}

```
extend sys { my_struct: my_struct_s;
 run() is also {
 print my_struct using hex; // just for reference
 var my_list: list of uint(bits:32);
 my_list = pack(packing.high, my_struct);  // stage 1, pack
 my_list = my_list.reverse(); // stage 2, reverse to restore original order
 print my_list using hex;
 };
};
```

{/code}

And here's what the output should look like, note that the list reflects the original bit order

```
my_struct = my_struct_s-@0: my_struct_s
---------------------------------------------  @temp
0       %byte1:                  0x12
1       %word1:                   0x3456
2       %byte2:                  0x78
3       %byte3:                  0x9a
4       %byte4:                  0xff
5       %data1:                  0x555
6       %dword1:                   0x57fffaaa
7       %data2:                  0x15
my_list =
0.      12345678
```

1. 9affaaaa
2. ffff5555

No actual running requested.
Checking the test ...
Checking is complete - 0 DUT errors, 0 DUT warnings.