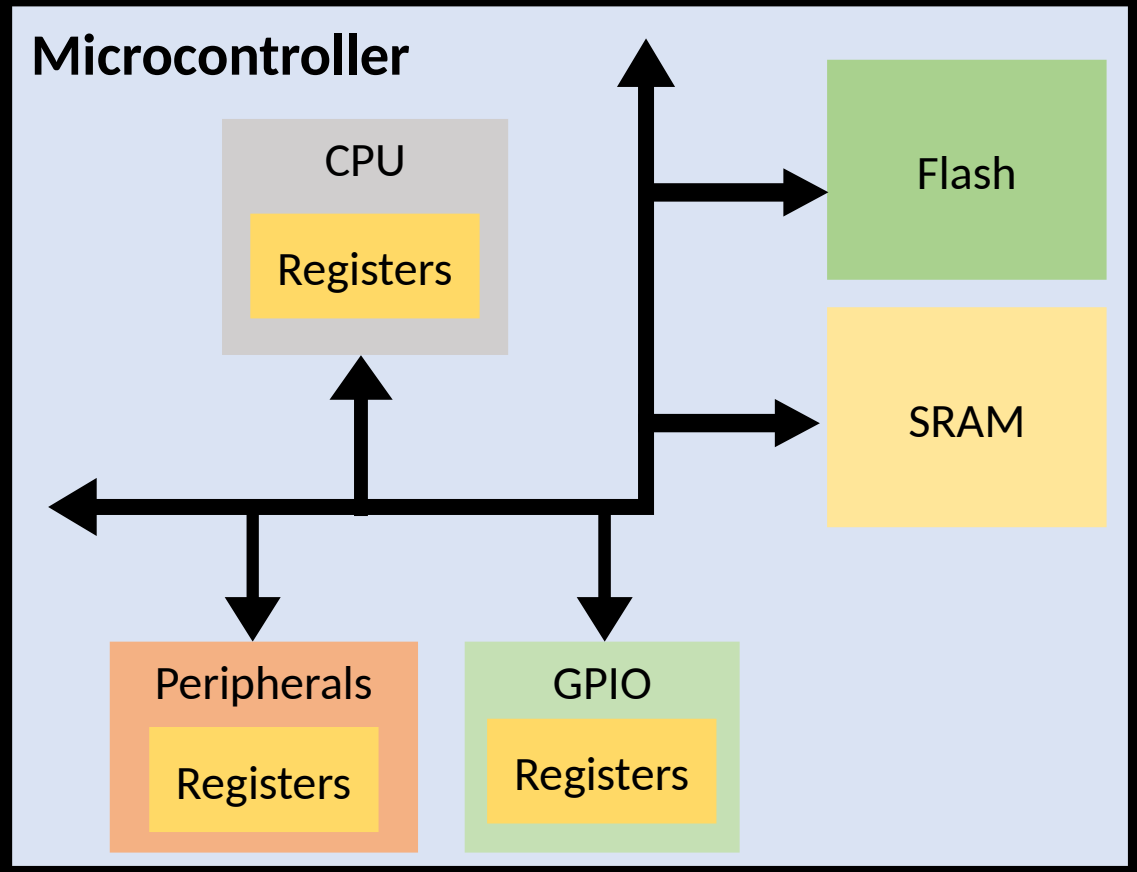
Memory Alignment

Embedded Software Essentials

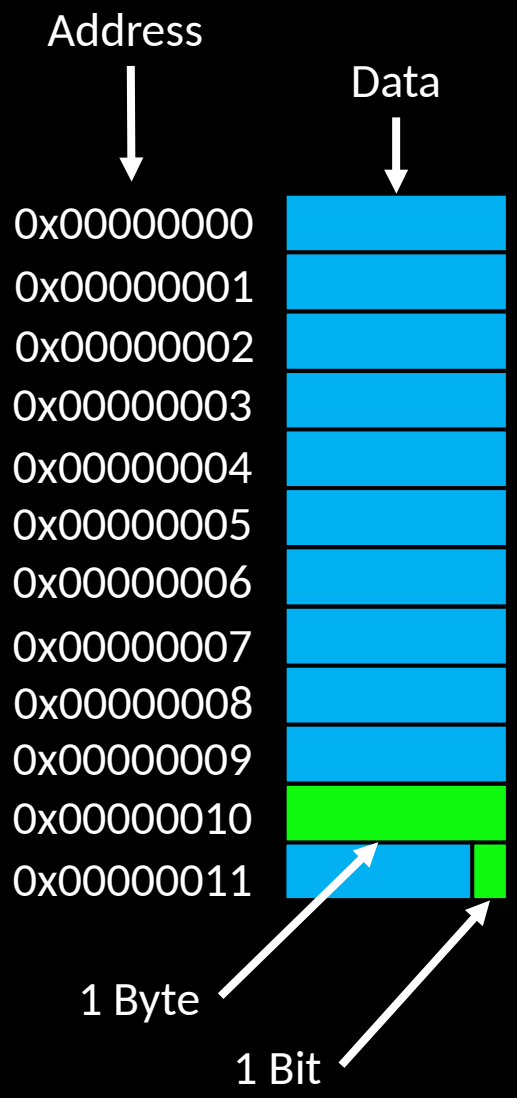
C2M1V5

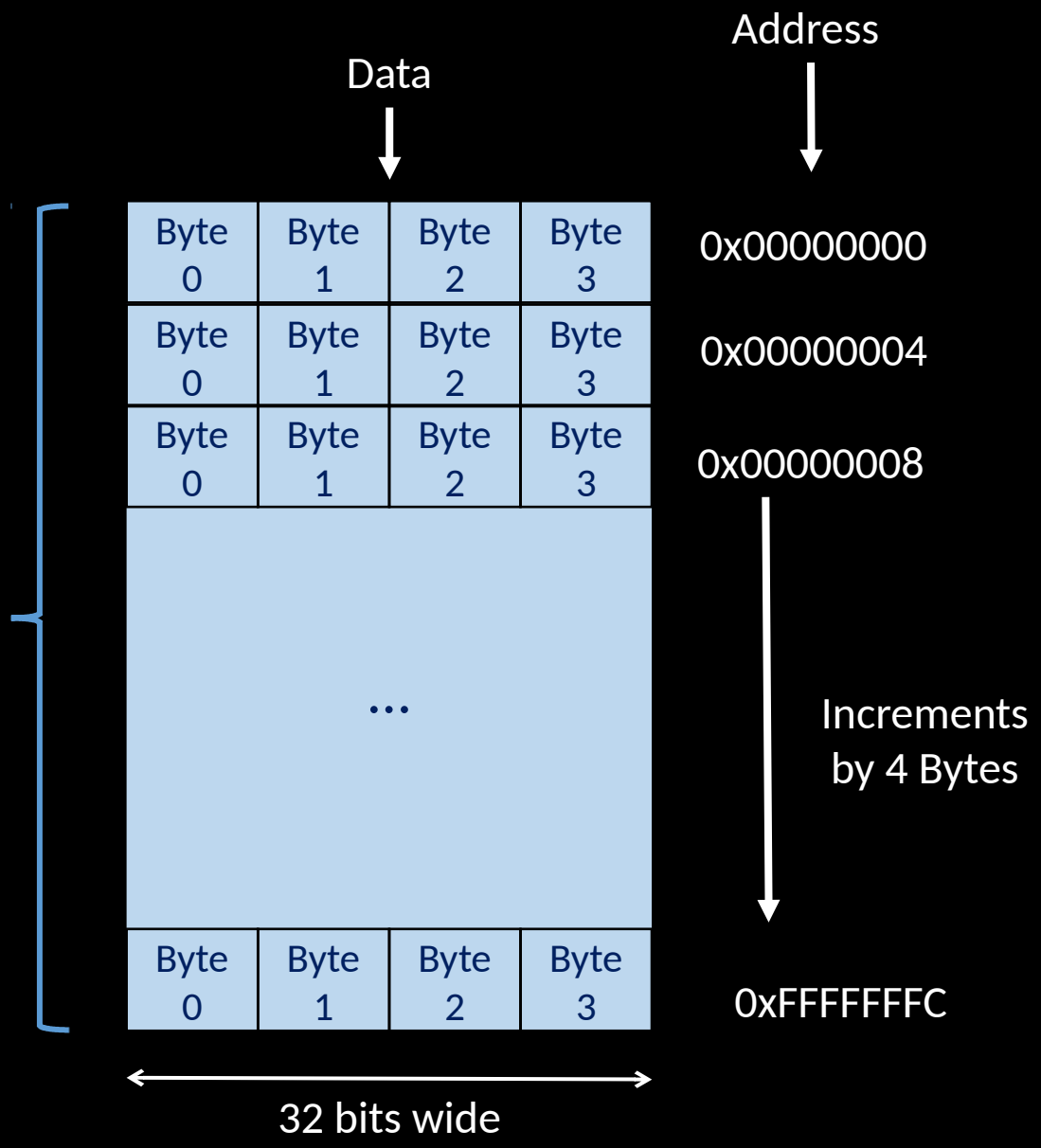
Memory [S1]

- Memory storage interacting with the CPU
  - Code Memory
  - Data Memory
  - Registers (Peripherals)
- Memory interfaces to CPU through Busses
- Load-Store architecture requires operations to occur in CPU
  - Data gets loaded into CPU
  - Data is operated on
  - Data is stored back

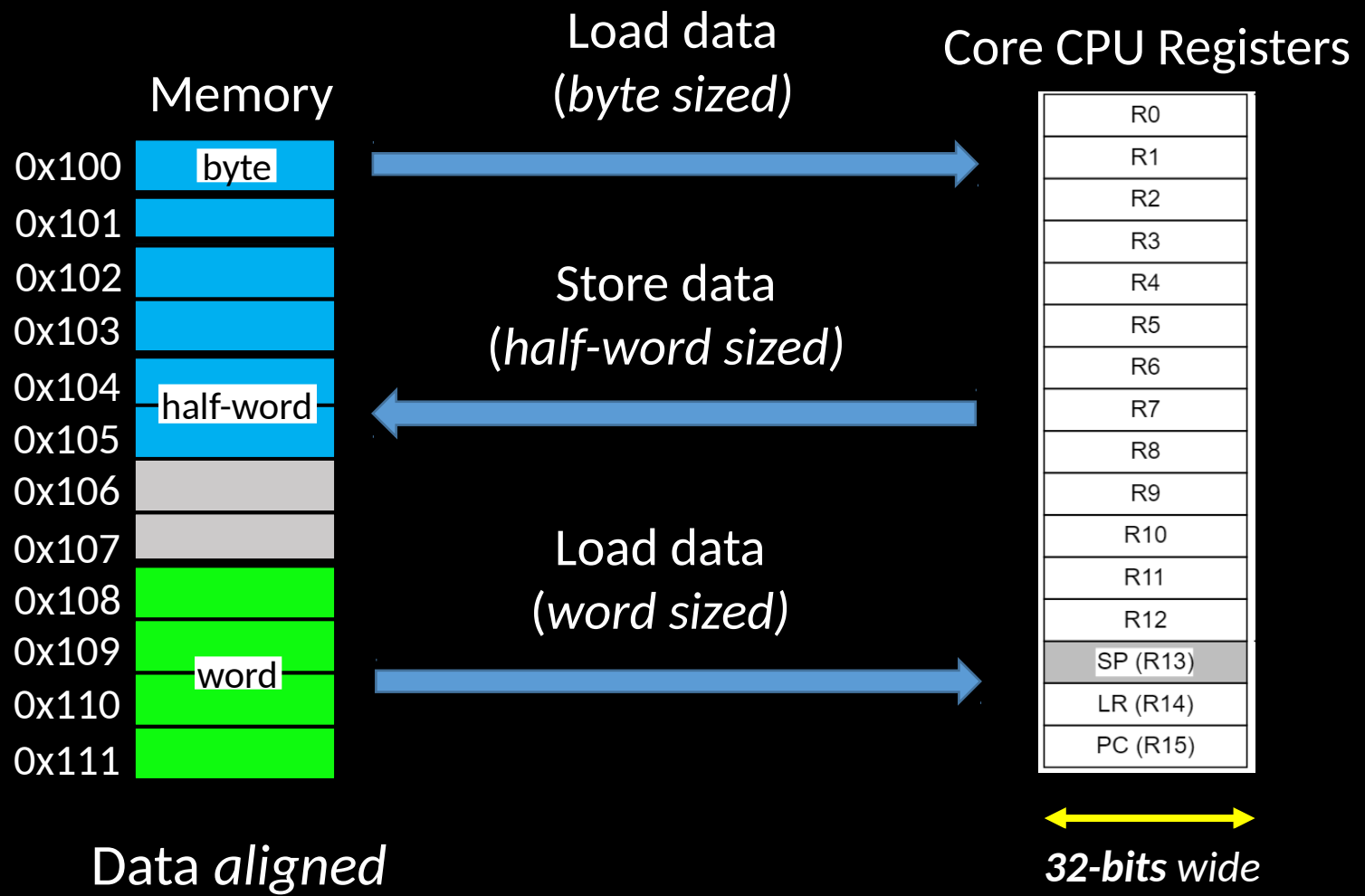


Memory Model [S2]





Memory Organization [S3]



Data *aligned*
CPU efficient
Memory inefficient

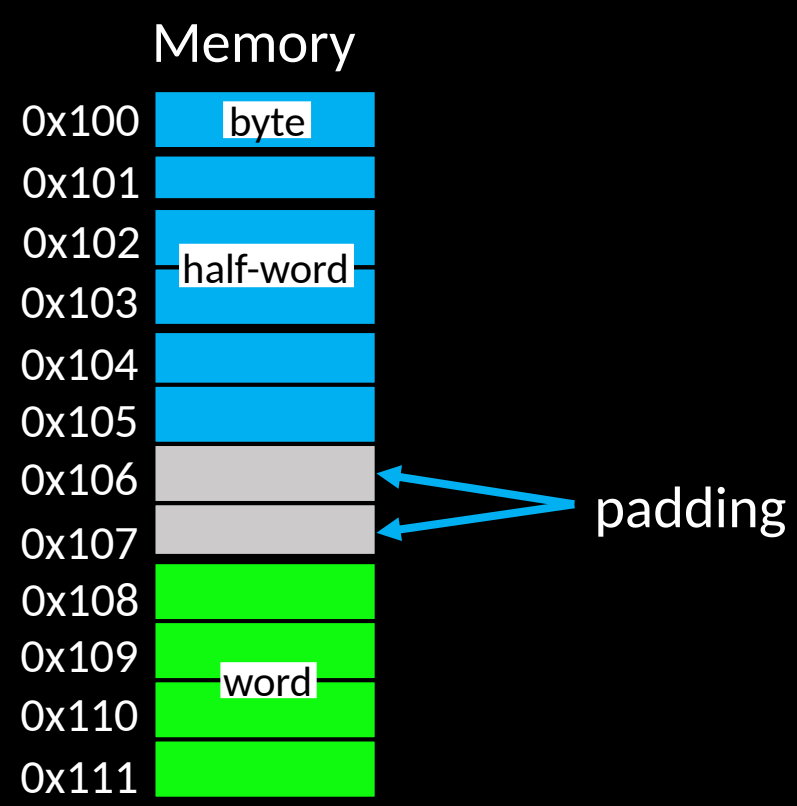
Assembly Load/Store Instructions

LDR – Load Word
STR – Store Word

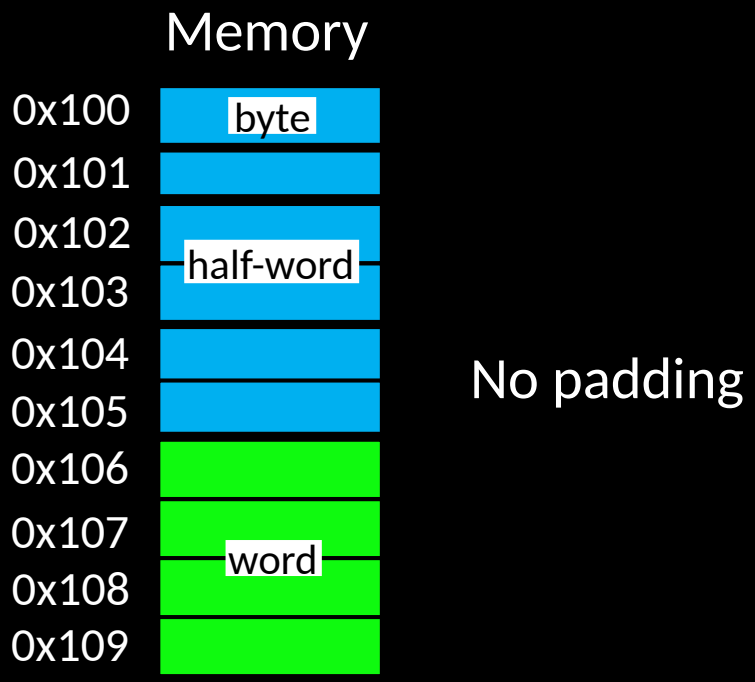
LDRH – Load Unsigned Word
LDRSH – Load Signed Half Word
STRH – Store Unsigned Half Word
STRSH – Store Signed Half Word

LDRB – Load Unsigned Byte
LDRSB – Load Signed Byte
STB – Store Unsigned Byte
STSB – Store Signed Byte

Memory Alignment [S4]

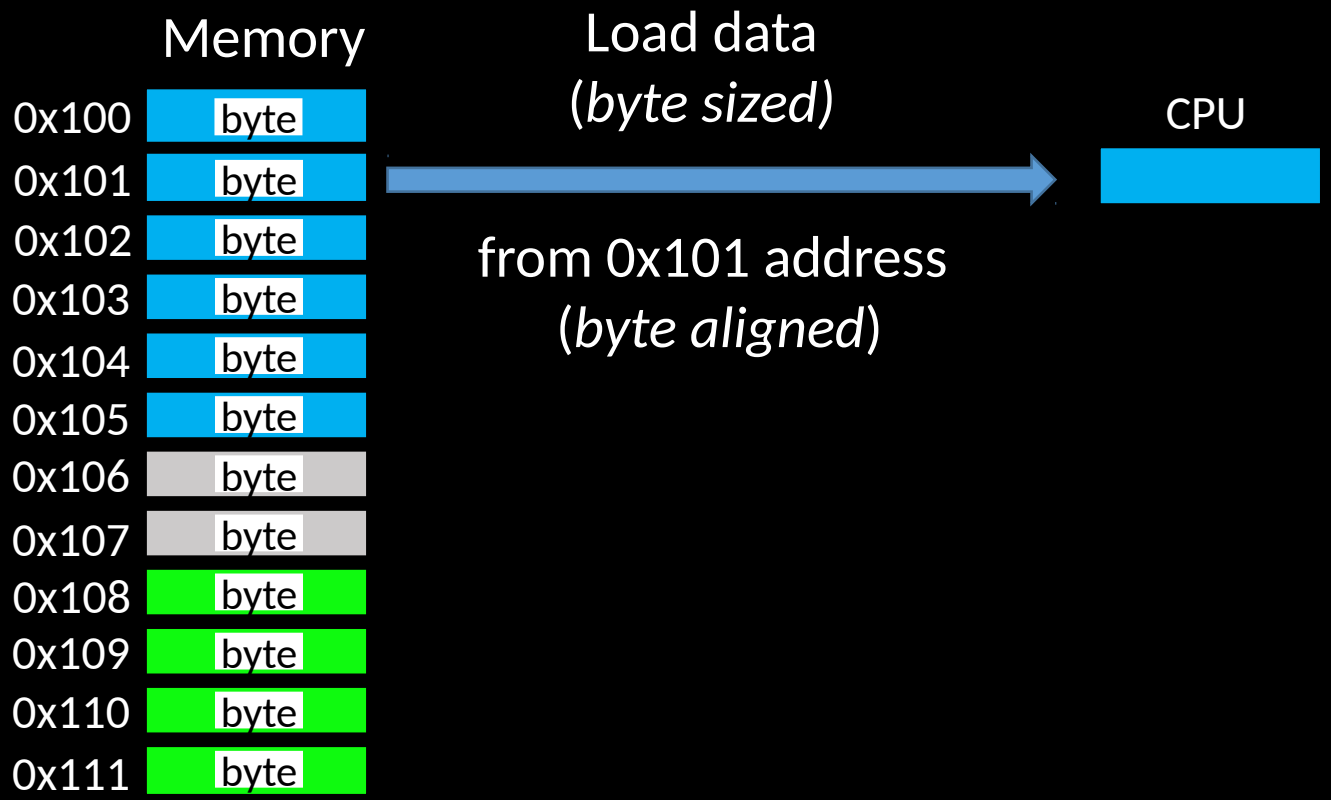


Data *aligned*
CPU efficient
Memory inefficient



Data *packed*
CPU inefficient
Memory efficient

Byte Load/Stores from Memory [S5]



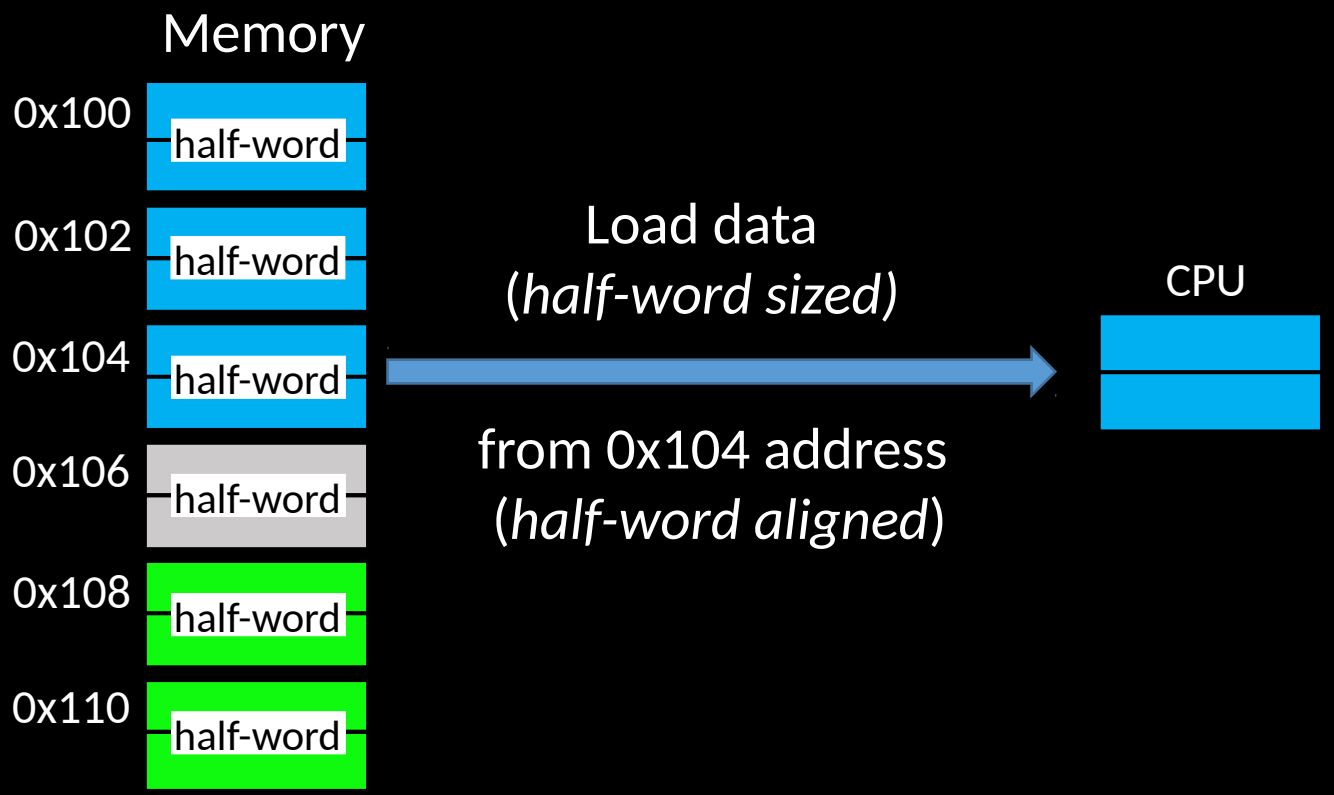
Load/store data occurs only at ***aligned addresses*** in memory

Data *aligned*

CPU ***efficient***

Memory inefficient

Half-Word Load/Stores from Memory [S6]



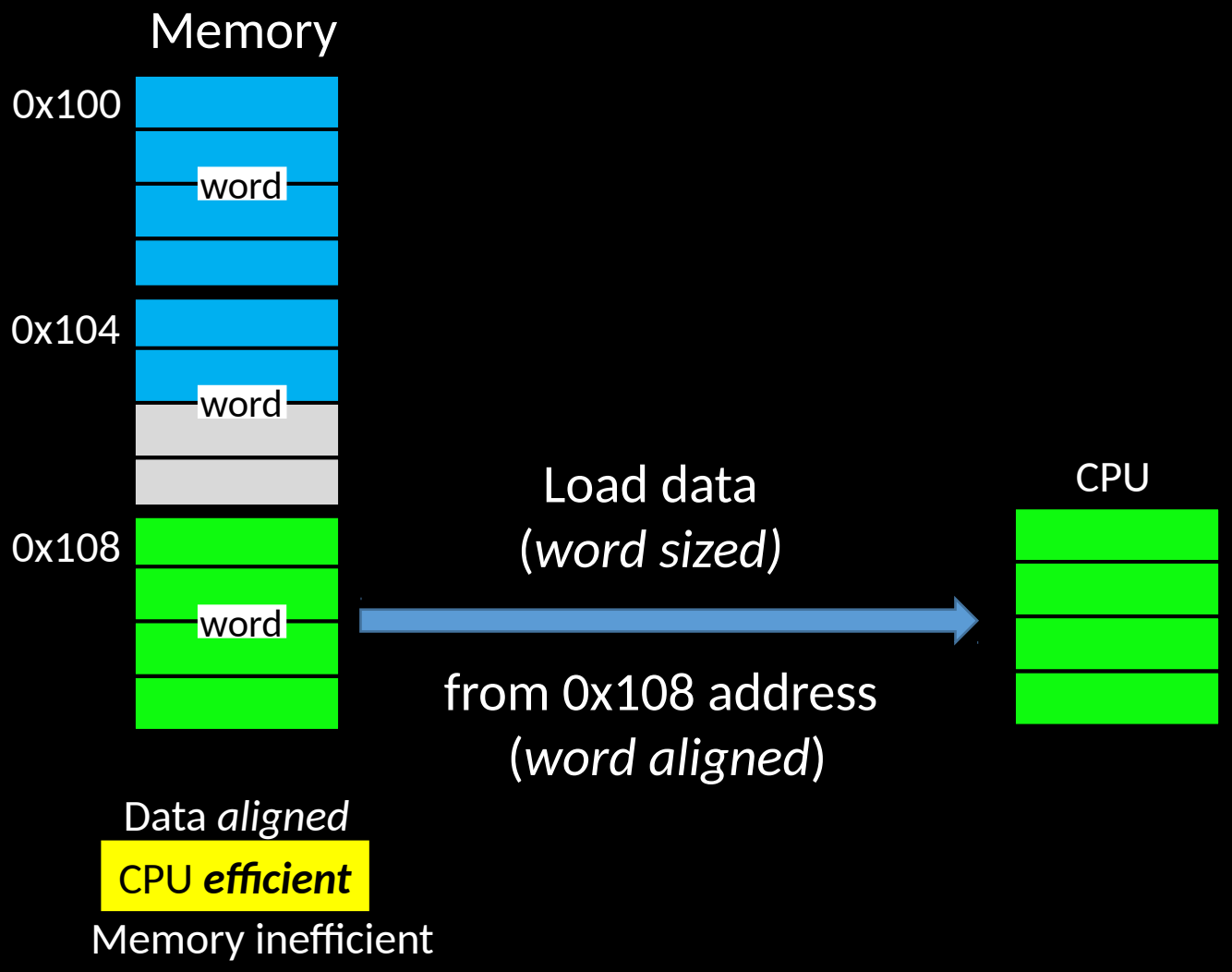
Load/store data occurs only at ***aligned addresses*** in memory

Data *aligned*

CPU ***efficient***

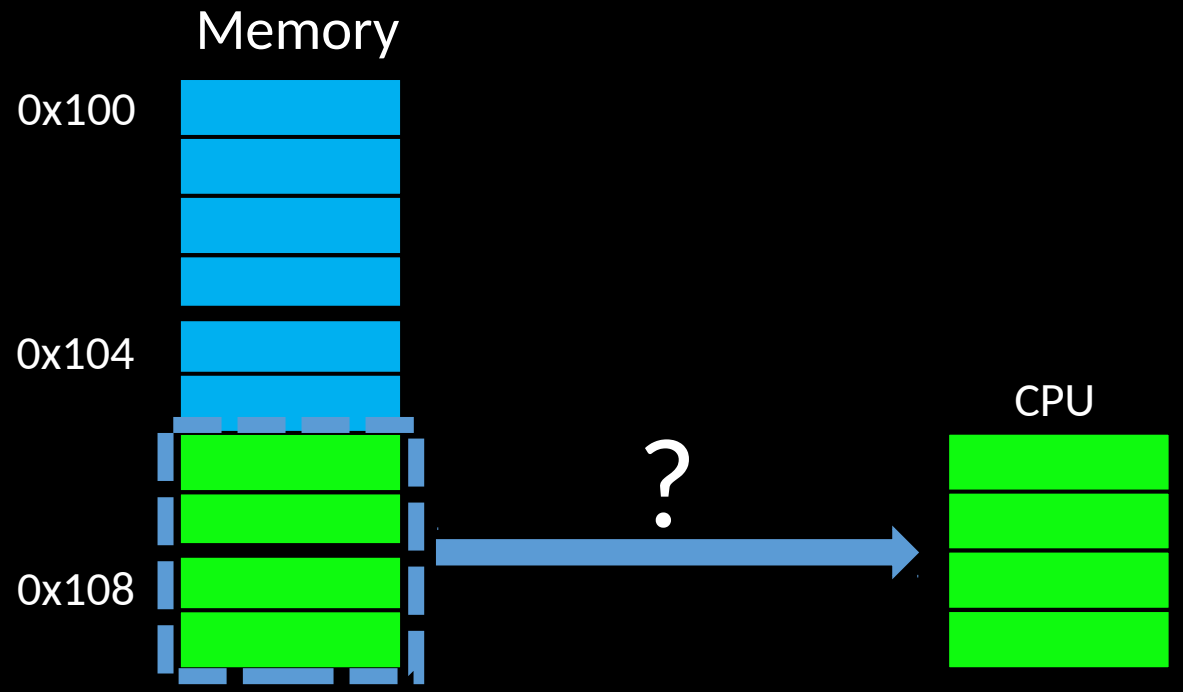
Memory inefficient

Word Load/Stores from Memory [S7]



Load/store data occurs only at ***aligned addresses*** in memory

Unaligned Access [S8]

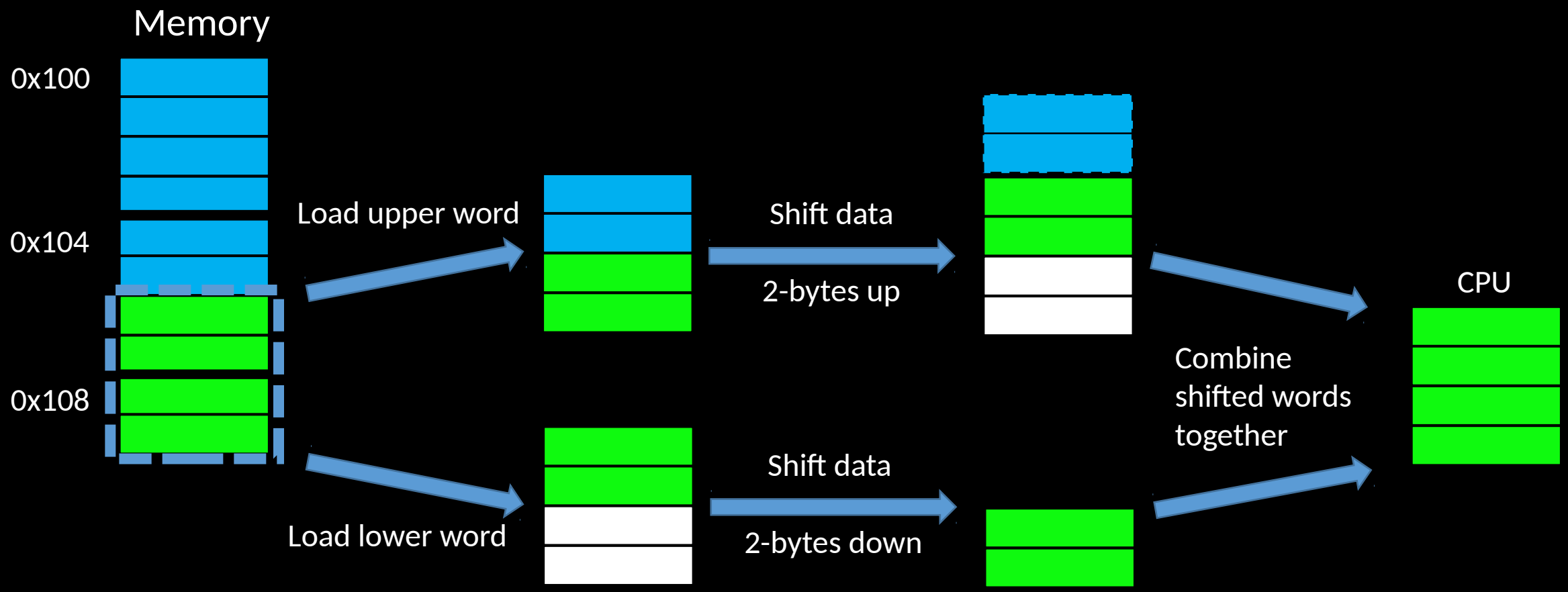


Data *packed*
CPU inefficient
Memory efficient

Load/store data occurs only at ***aligned addresses*** in memory

How to load **misaligned data**?

Unaligned Word Example [S9]



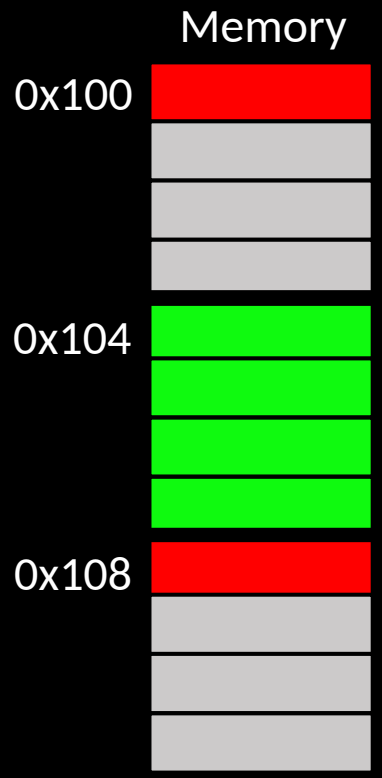
Data *packed*

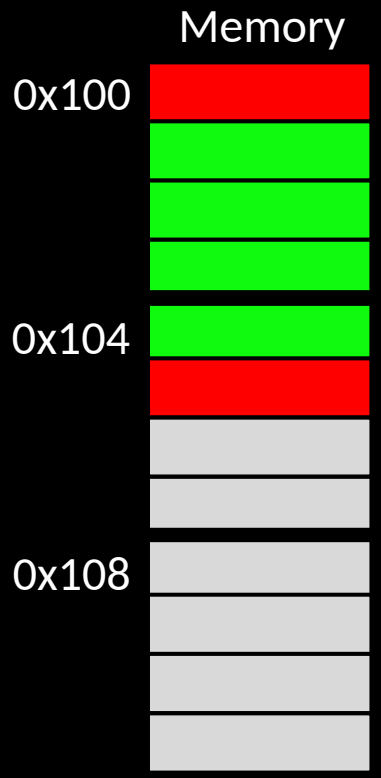
CPU ***inefficient***

Memory efficient

Struct in Memory [S10a]

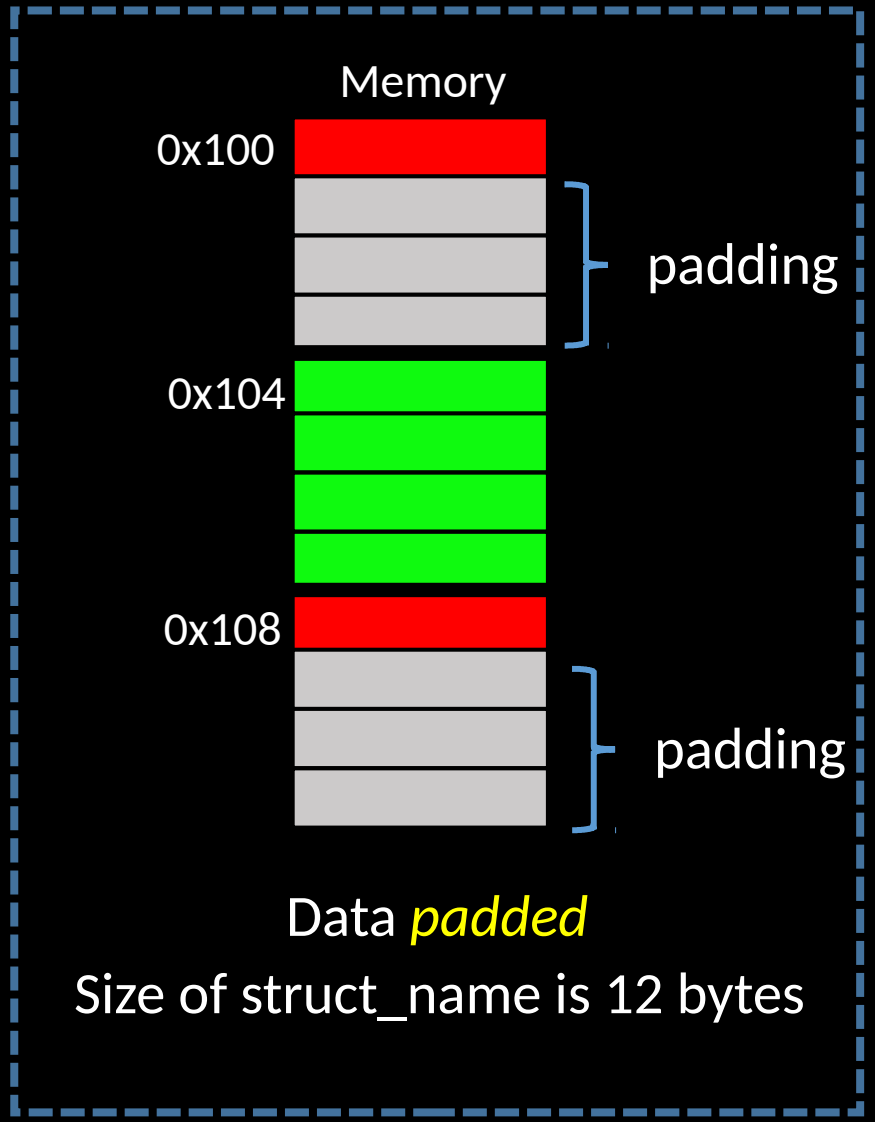
```
struct struct_name {
   char  var1;
   int     var2;
   char  var3;
};
```

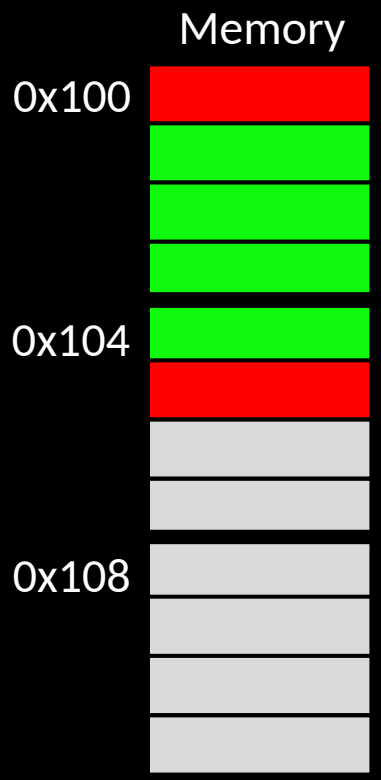




Struct in Memory [S10b]

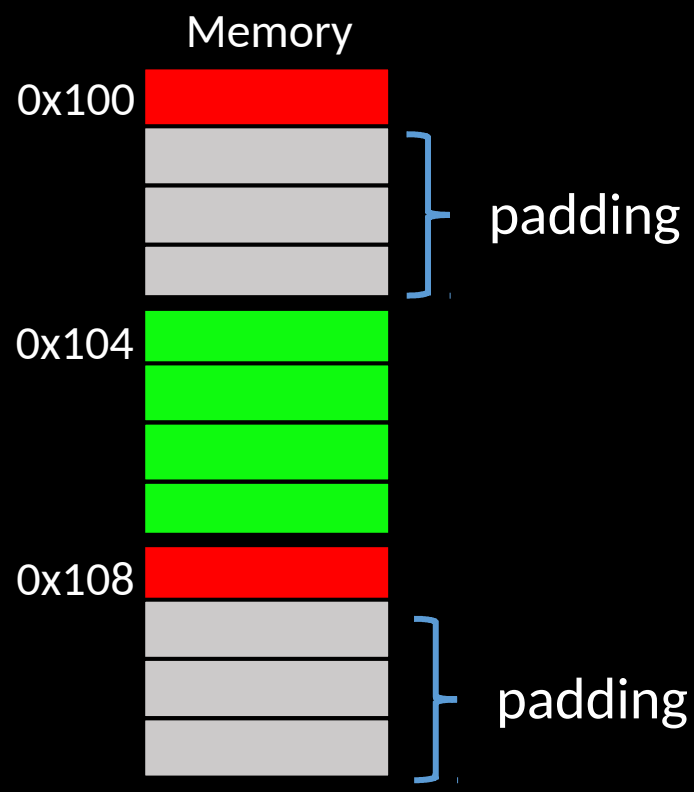
```
struct struct_name {
   int_fast8_t    var1;
   int_fast16_t   var2;
   int_fast8_t    var3;
};
```





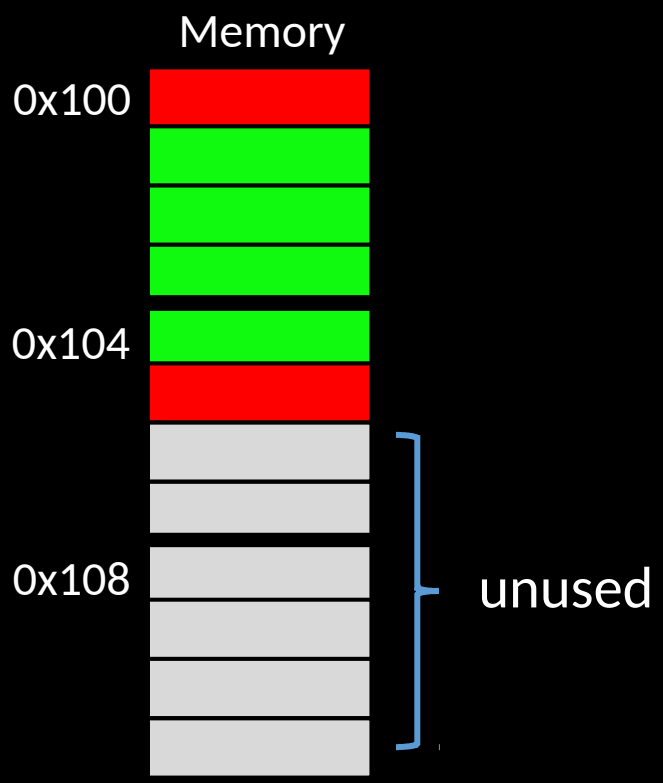
Struct in Memory [S10c]

```
struct struct_name {
   int8_t   var1;
   int32_t var2;
   int8_t   var3;
} __attribute__ ((packed));
```



Data *padded*

Size of struct_name is 12 bytes



Data *packed*

Size of struct_name is 6 bytes