Load and Relocation Algorithm

Input: MERL file
Output: MIPS program, loaded and relocated into RAM

```
read()   // skip cookie
endMod <- read()   // end of MERL file
codeLen <- read() - 12   // subtract header (12 bytes) => length of code only

alpha <- findFreeRAM(codeLen)

for (i = 0; i < codeLen; i += 4)
   MEM[alpha + i] <- read()

i <- codeLen + 12   // start of relocation table

while (i < endMod) {
   format <- read()
   if (format == 1)
      rel <- read()   // addr to be relocated, relative to start of header
                      //   not start of code
      MEM[alpha + rel - 12] += alpha - 12   // adjust forward by begin_addr
                                            //   back by 12 ~ header not included
   else
      ERROR
   i += 8
```

Linker Algorithm

Input: MERL files M_1 and M_2
Output: Single MERL file with M_2 linked after M_1

```
alpha <- M1.codeLen - 12

relocate M2.code by alpha

add alpha to every address in M2.symbol

if M1.exports.labels INTERSECT M2.exports.labels != NULL then ERROR

// Try to match labels imported in M1 that are exported from M2
for each (addr1, label) in M1.imports
   if (addr2, label) is in M2.exports
      M1.code[addr1] <- addr2
      remove (addr1, label) from M1.imports
      add addr1 to M1.relocates

// Try to match labels imported in M2 that are exported from M1
for each (addr2, label) in M2.imports
   if (addr1, label) is in M1.exports
      M1.code[addr2] <- addr1
      remove (addr2, label) from M2.imports
      add addr2 to M2.relocates

// Combine remaining imports, exports, relocates
imports = M1.imports UNION M2.imports
exports = M1.exports UNION M2. exports
relocates = M1.relocates UNION M2.relocates

Output: MERL Cookie
        total codeLen + total(imports, exports, relocate) + 12
        total codeLen + 12
        M1.code
        M2.code
        imports, exports, relocates
```