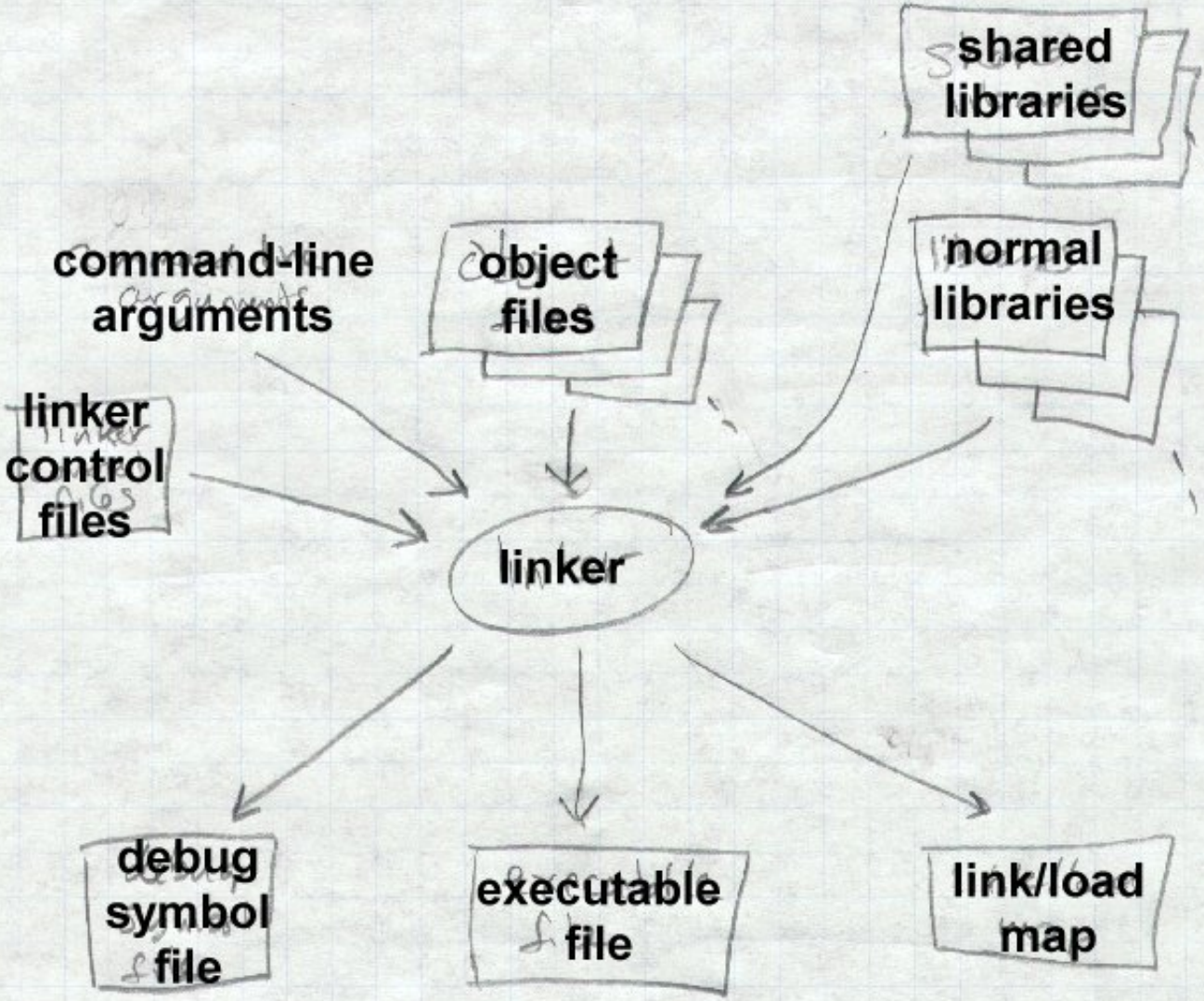
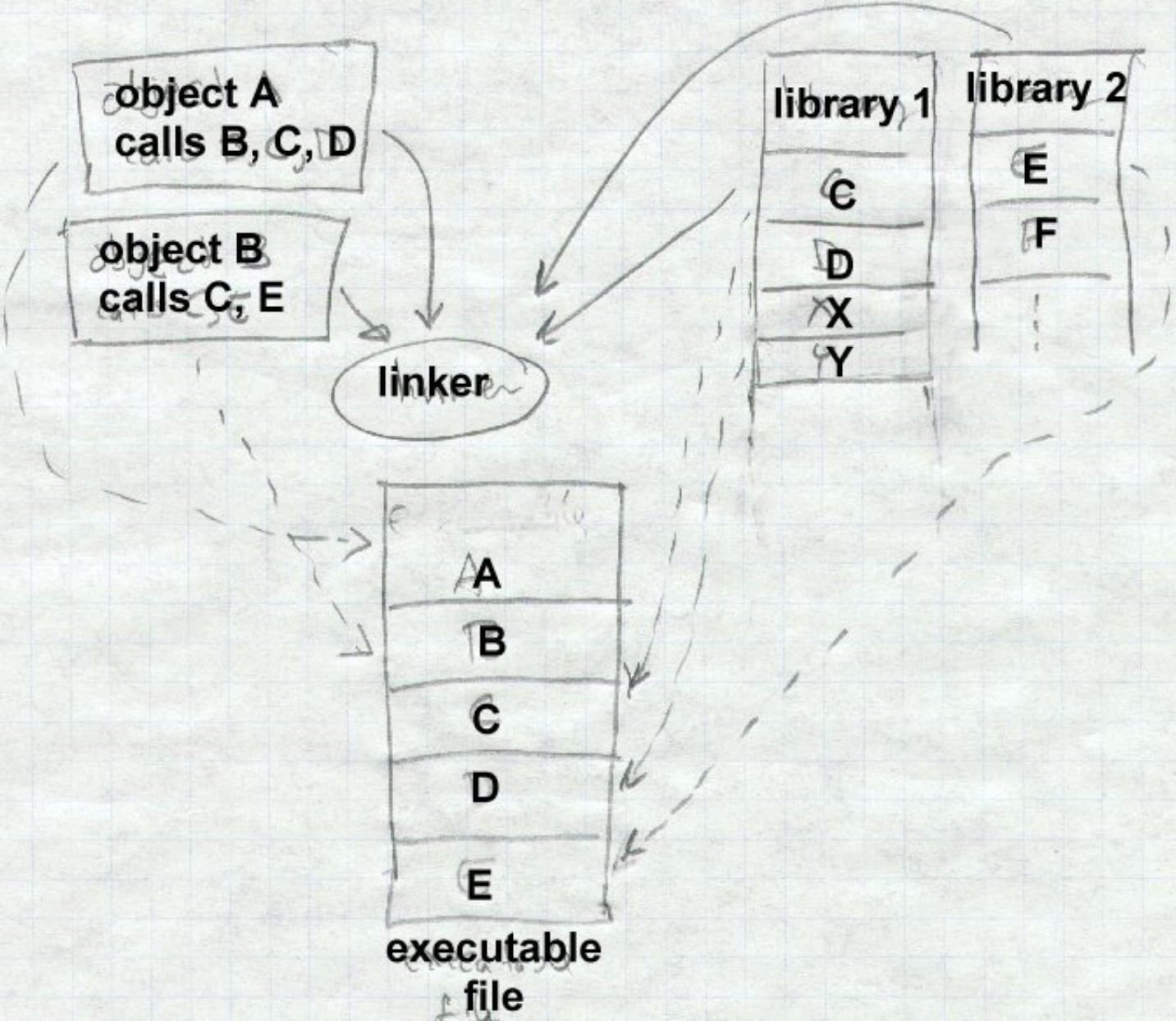
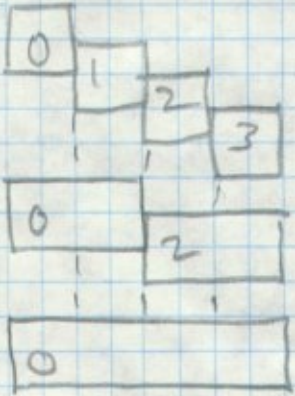

shared libraries
command-line arguments
object files
normal libraries
linker control files
linker
debug symbol file
executable file
link/load map


object A
calls B, C, D
object B
calls C, E
linker
library 1
C
D
X
Y
library 2
E
F
A
B
C
D
E
executable
file

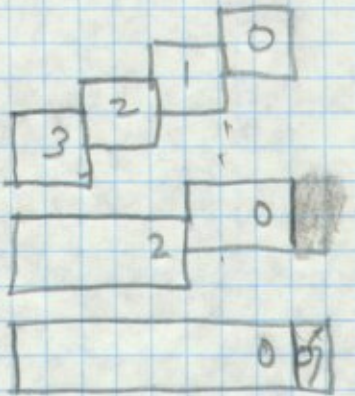


bytes

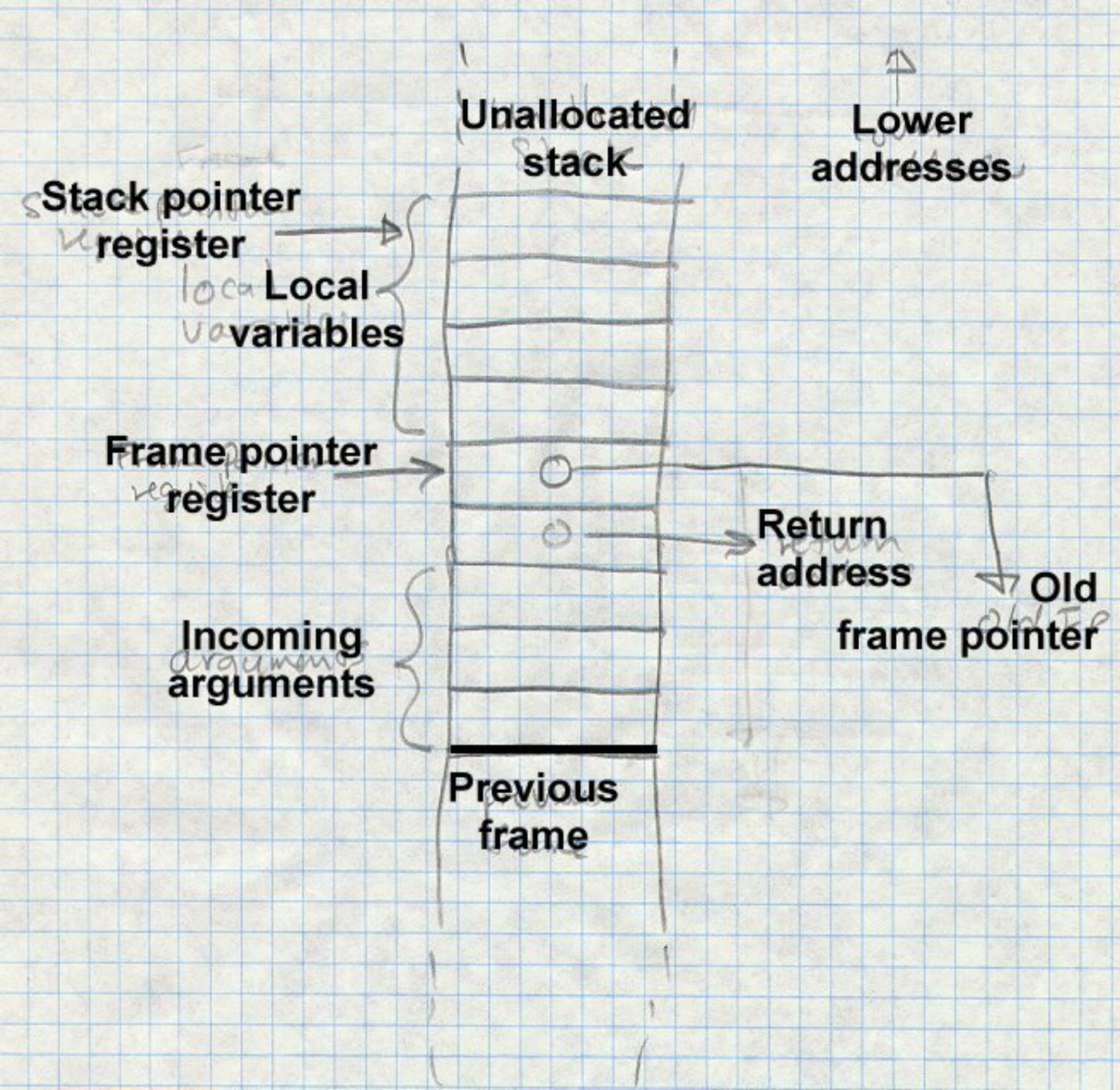
16-bit words

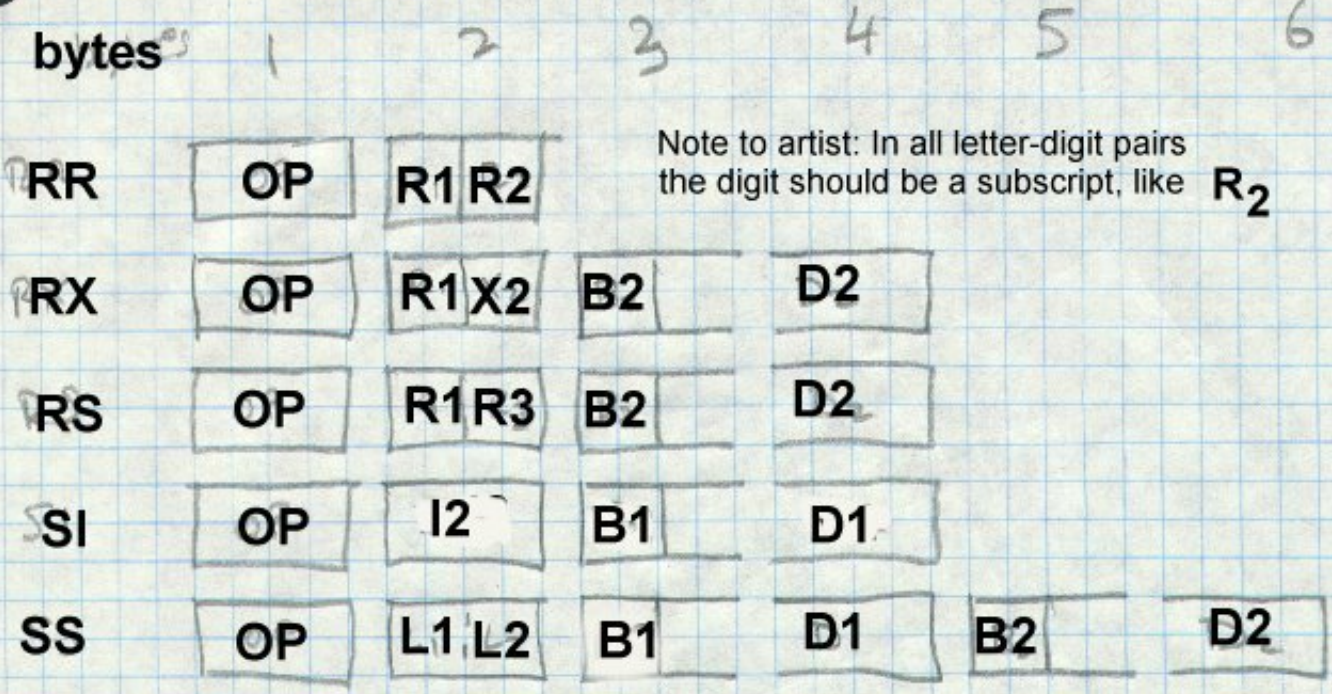
32-bit words

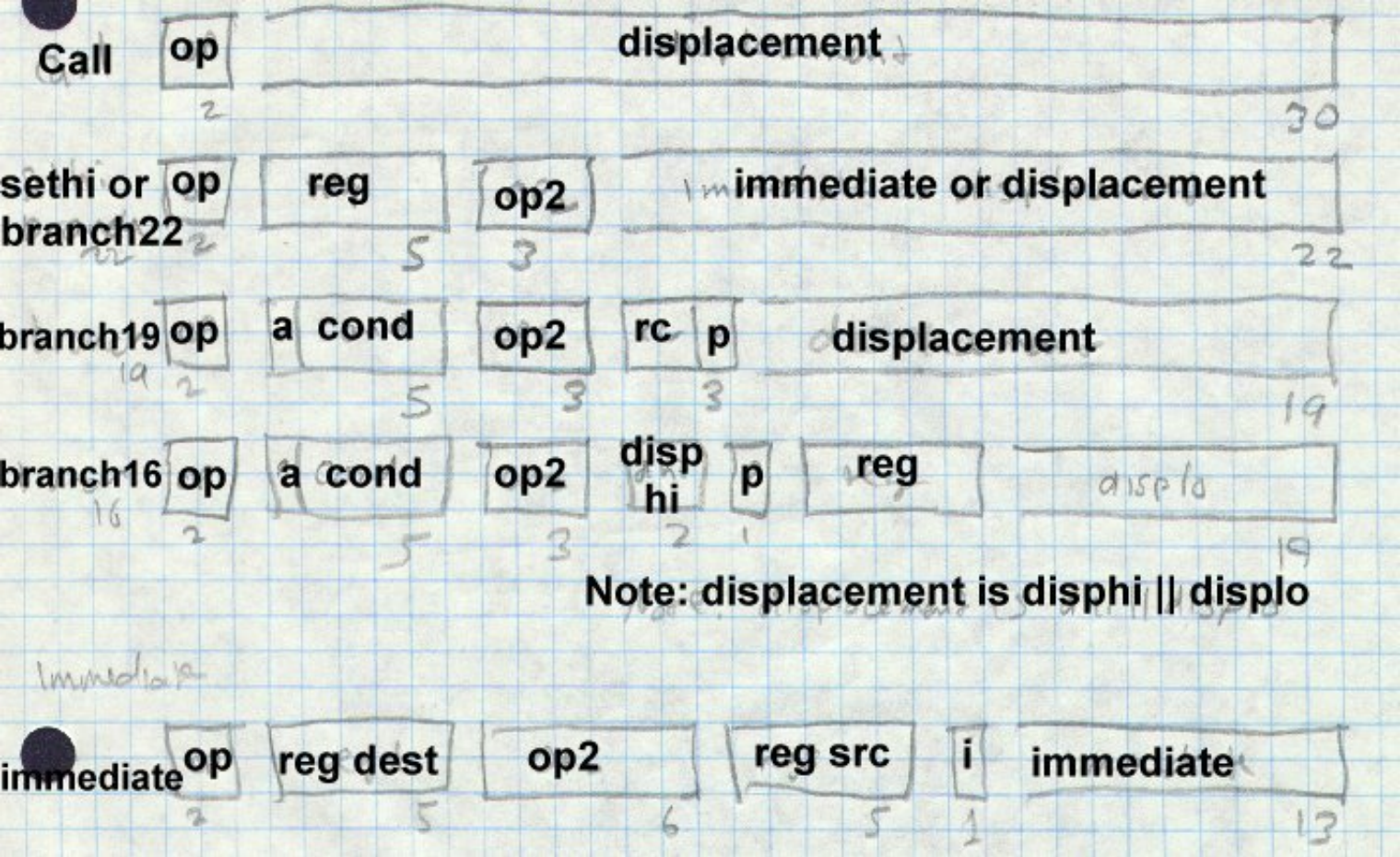
big-endian

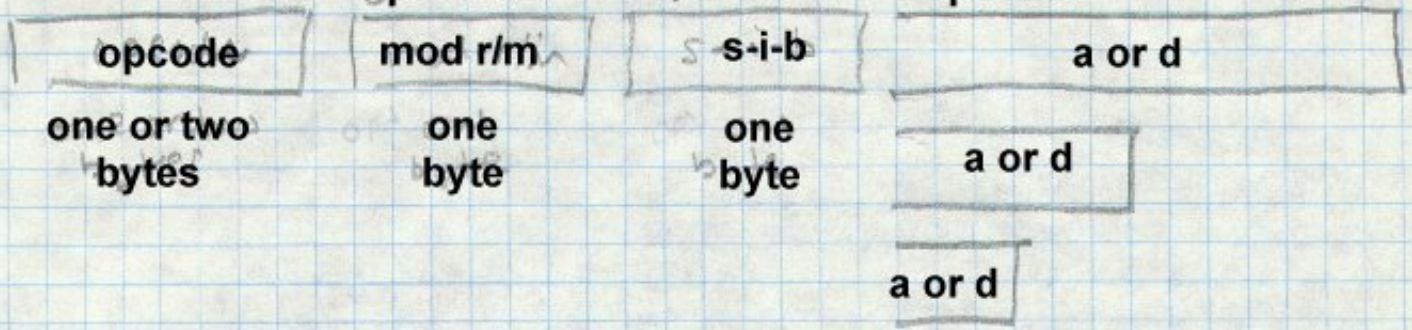


little-endian


Unallocated stack
Lower addresses
Stack pointer register
Local variables
Frame pointer register
Return address
Old frame pointer
Incoming arguments
Previous frame


bytes
1
2
3
4
5
6
RR
OP
R1 R2
Note to artist: In all letter-digit pairs the digit should be a subscript, like R_2
RX
OP
R1 X2
B2
D2
RS
OP
R1 R3
B2
D2
SI
OP
I2
B1
D1
SS
OP
L1 L2
B1
D1
B2
D2

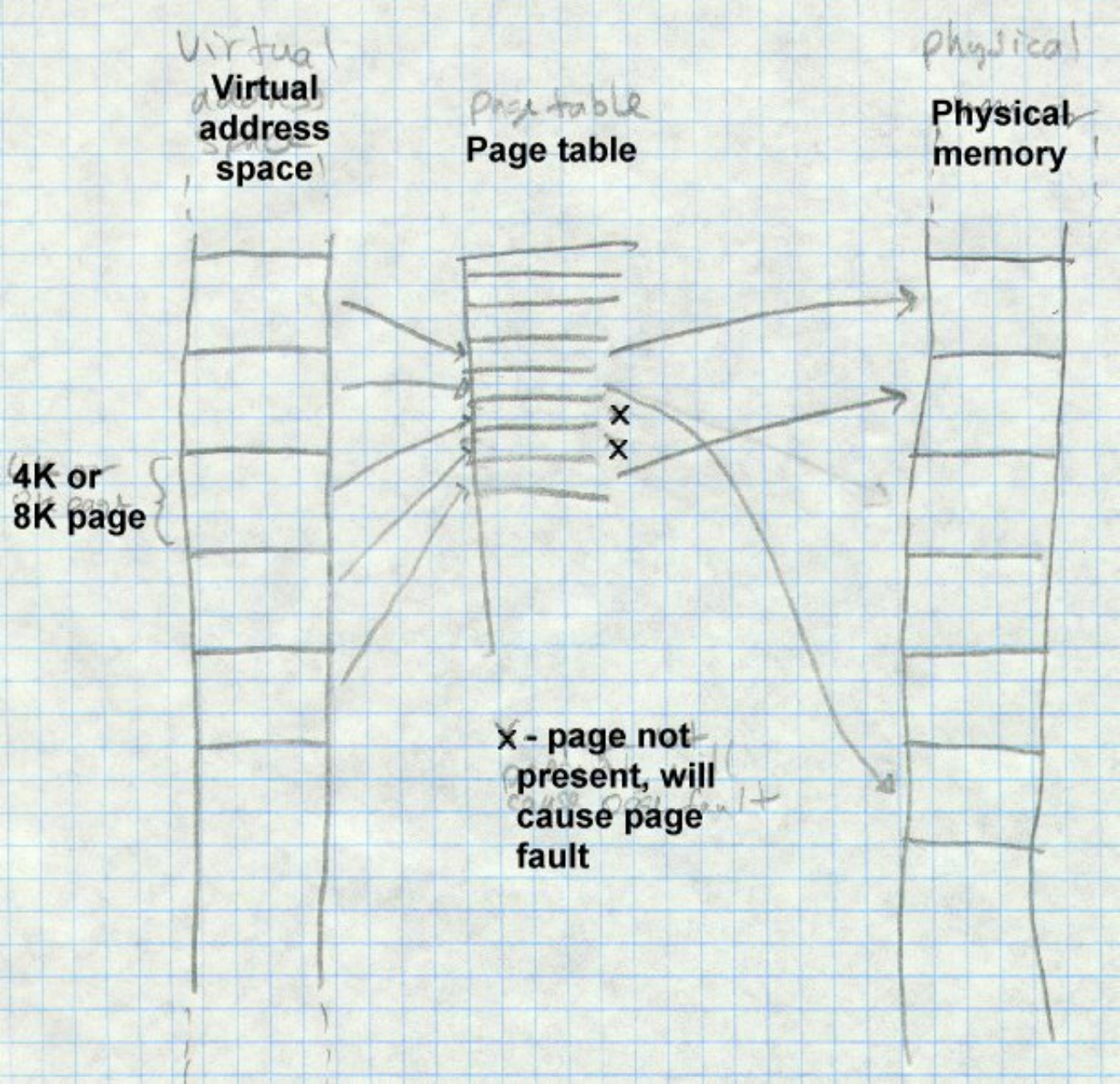

Call
op
2
displacement
30
sethi or branch22
op
2
reg
5
op2
3
immediate or displacement
22
branch19
op
2
a
cond
5
op2
3
rc
p
3
displacement
19
branch16
op
2
a
cond
5
op2
3
disp hi
2
p
1
reg
displo
19
Note: displacement is disphi || displo
immediate
op
2
reg dest
5
op2
6
reg src
5
i
1
immediate
13

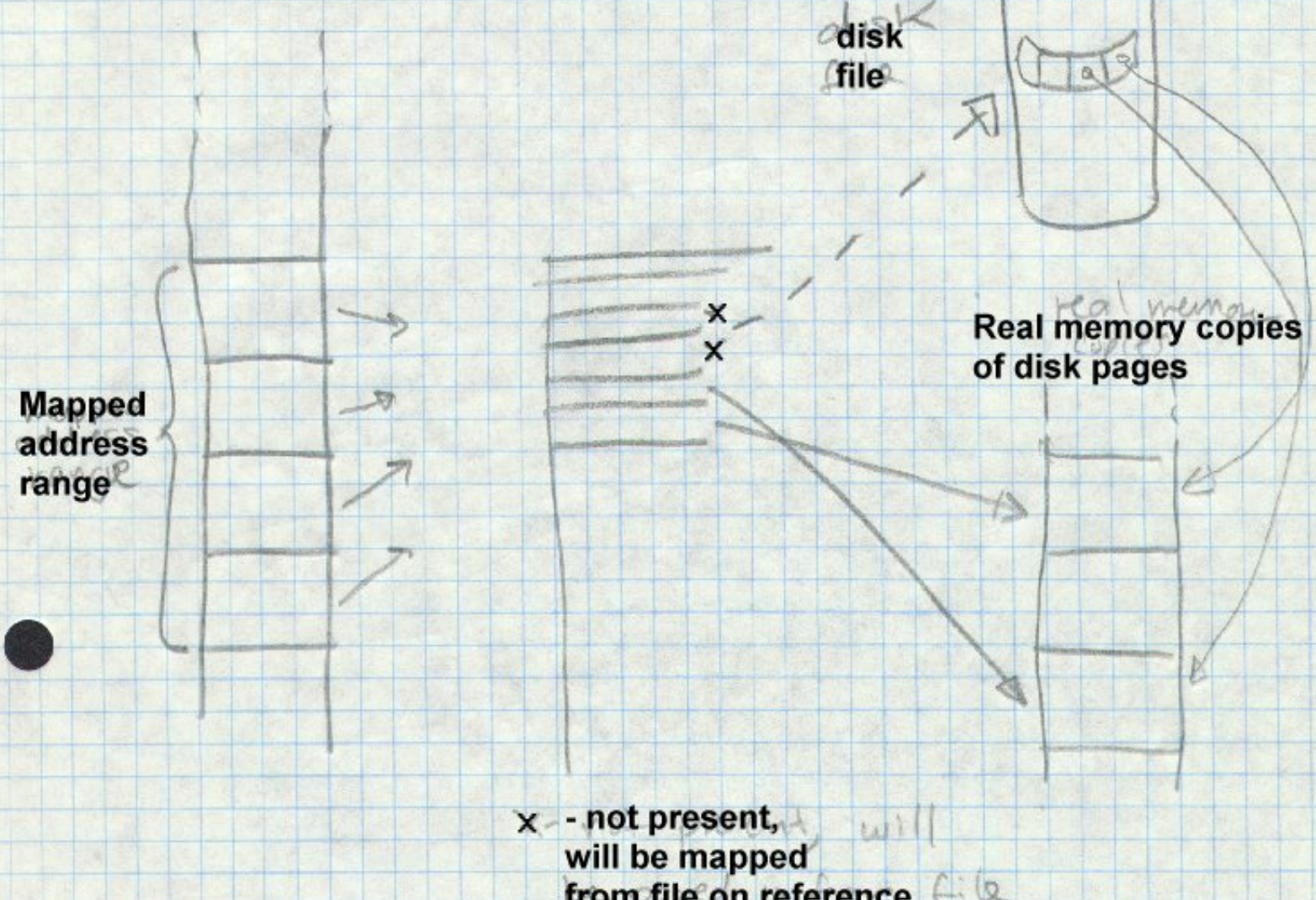


Mod r/m specifies address format

S-I-B specifies scaled index and/or base register

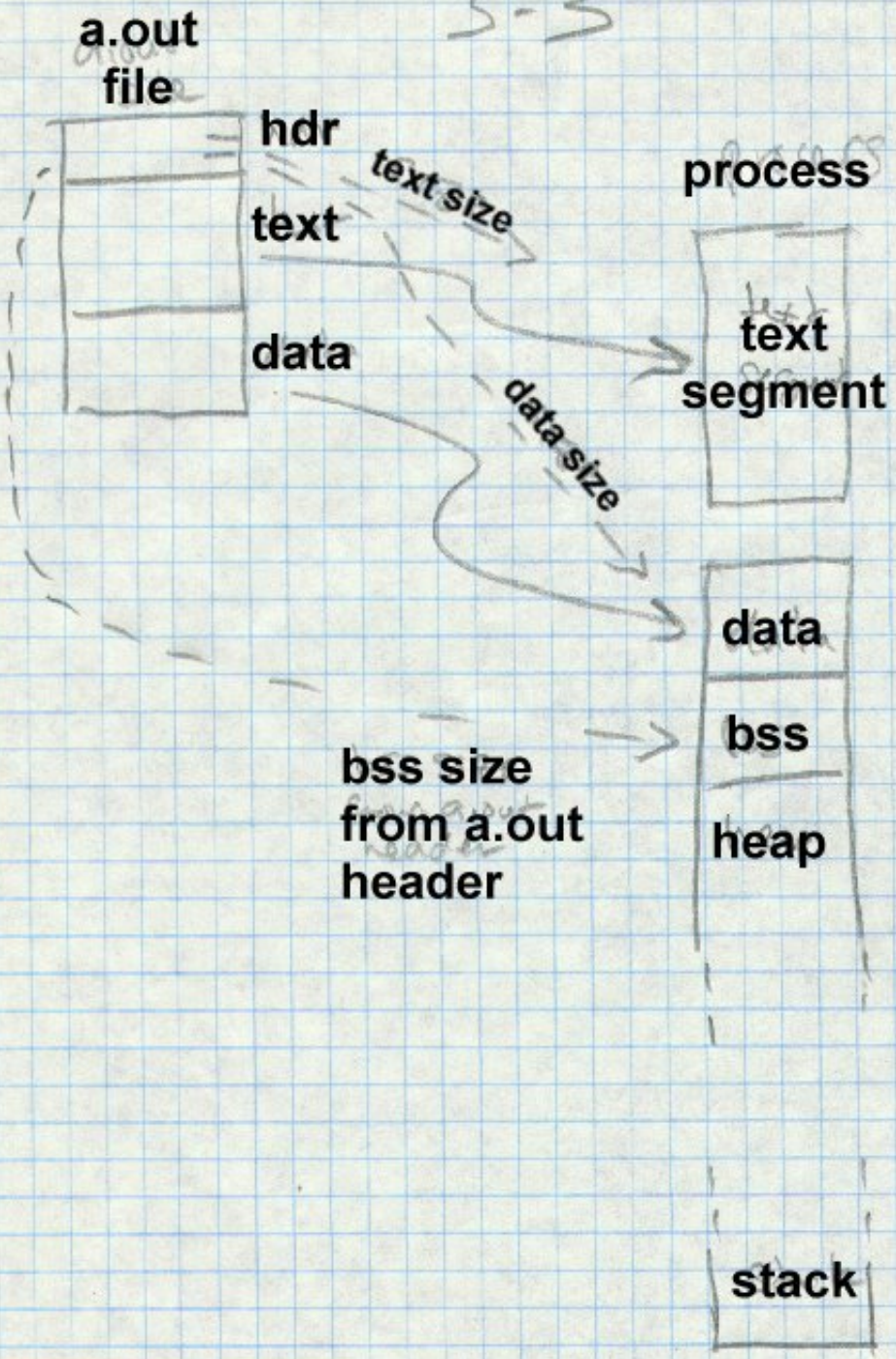
Address may be absolute or relative to base and/or index

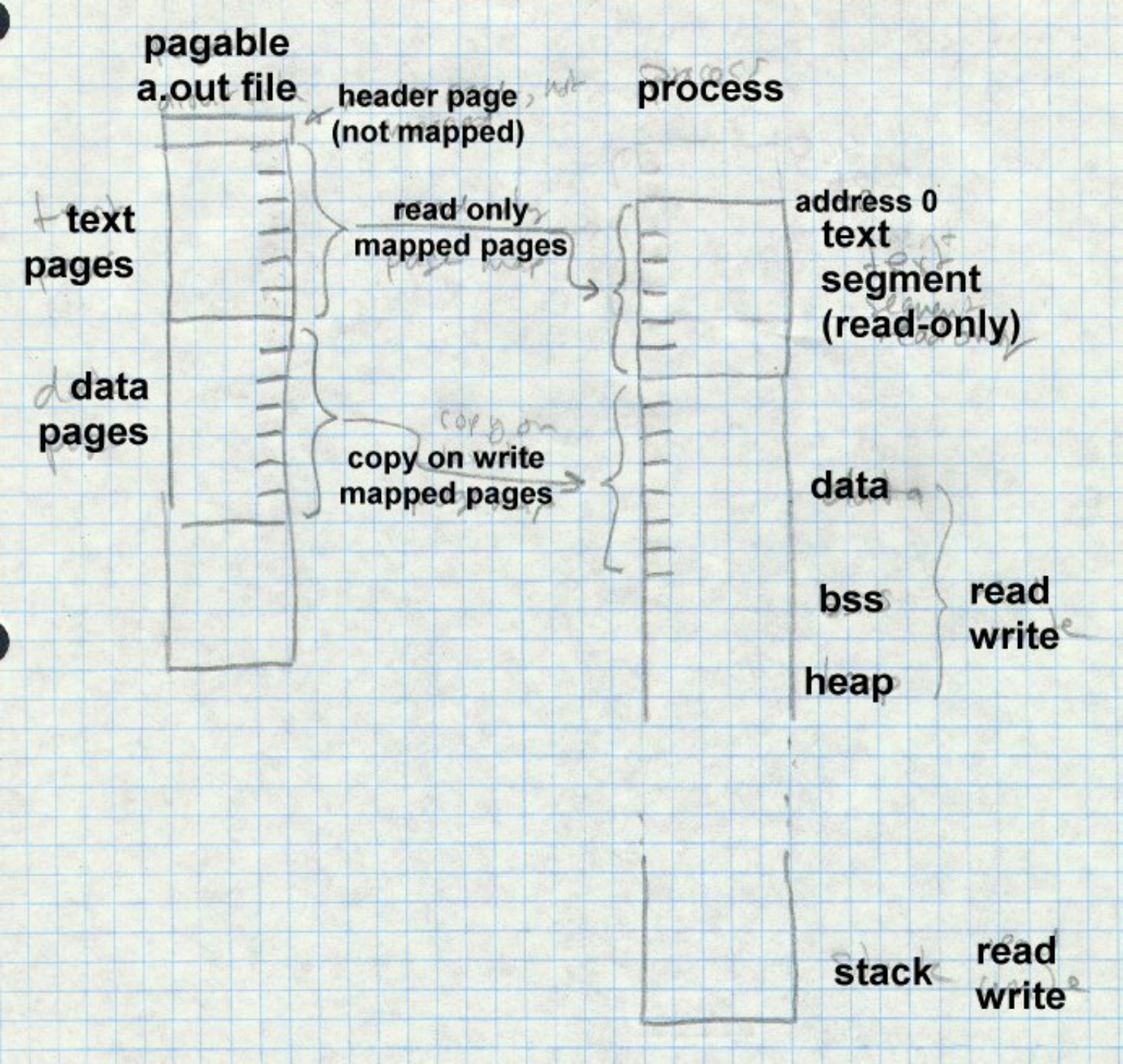

Virtual address space
Page table
Physical memory
4K or 8K page
x
x
x - page not present, will cause page fault

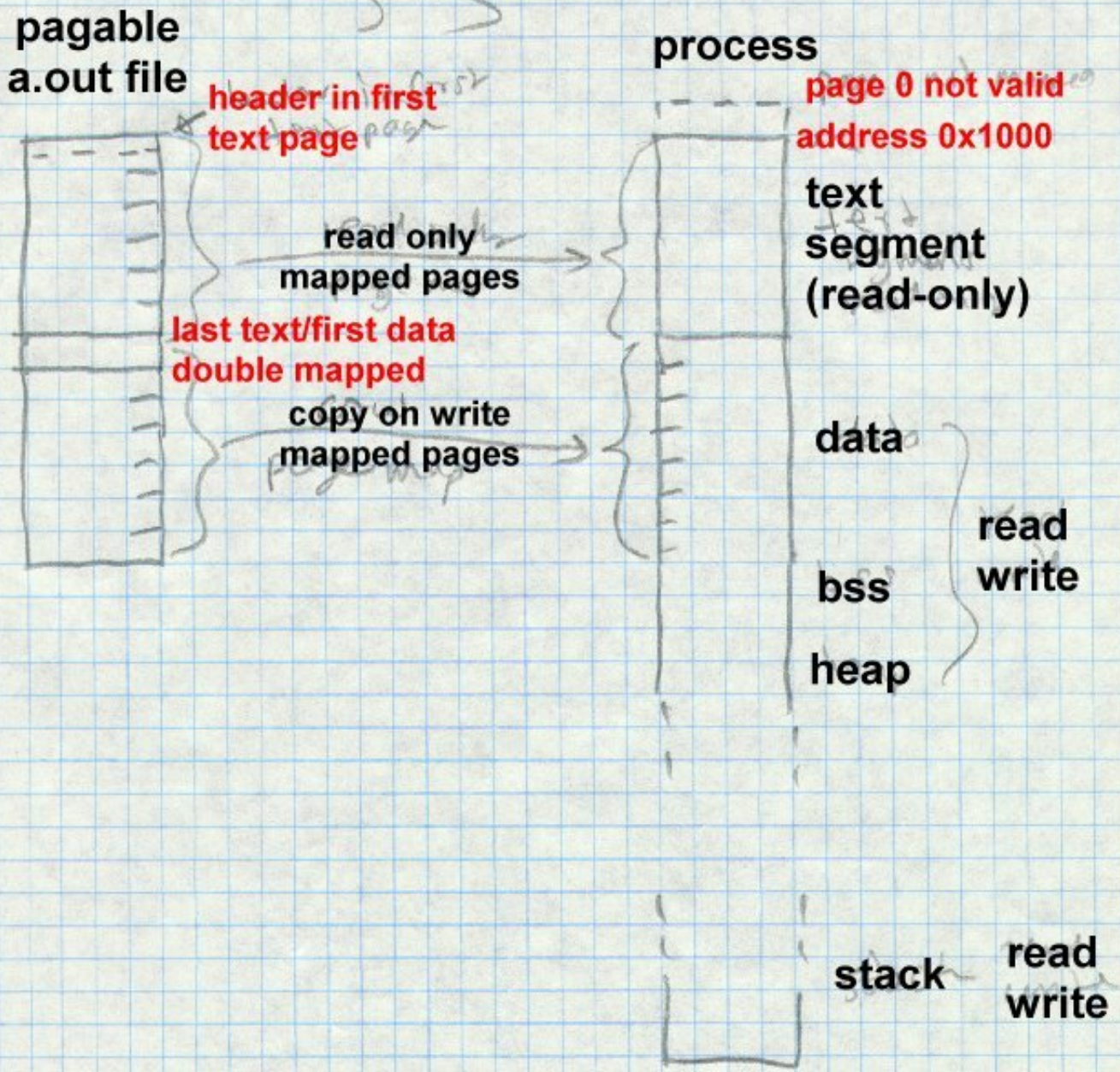

disk
file
Mapped
address
range
x
x
Real memory copies
of disk pages
x - not present,
will be mapped
from file on reference


a.out header
text section
data section
other sections

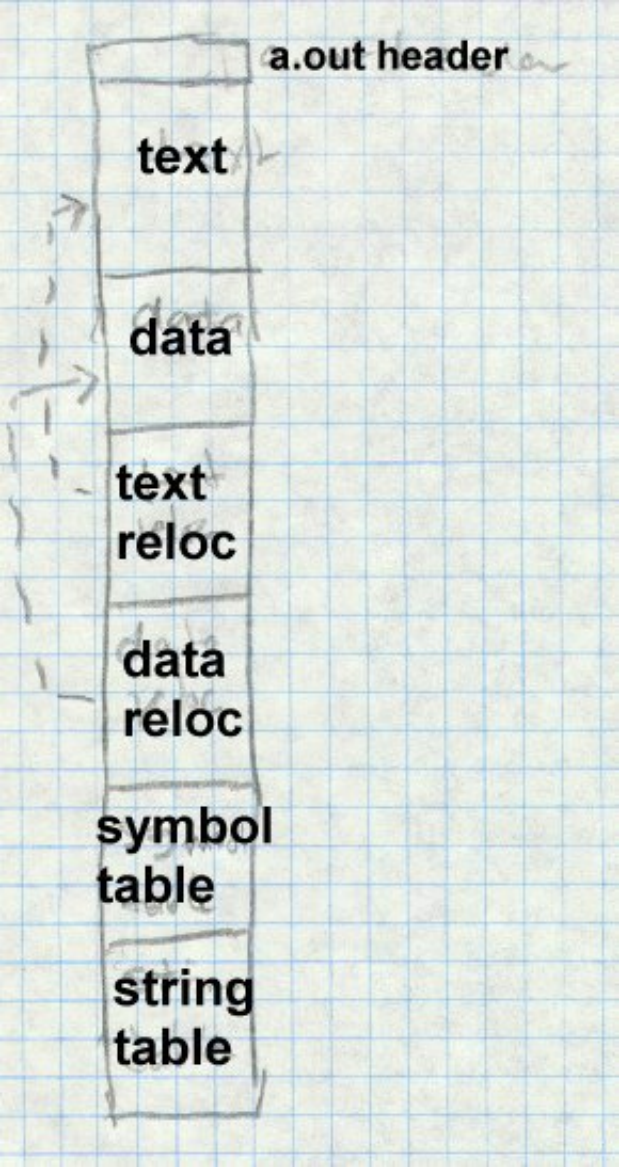
3-3

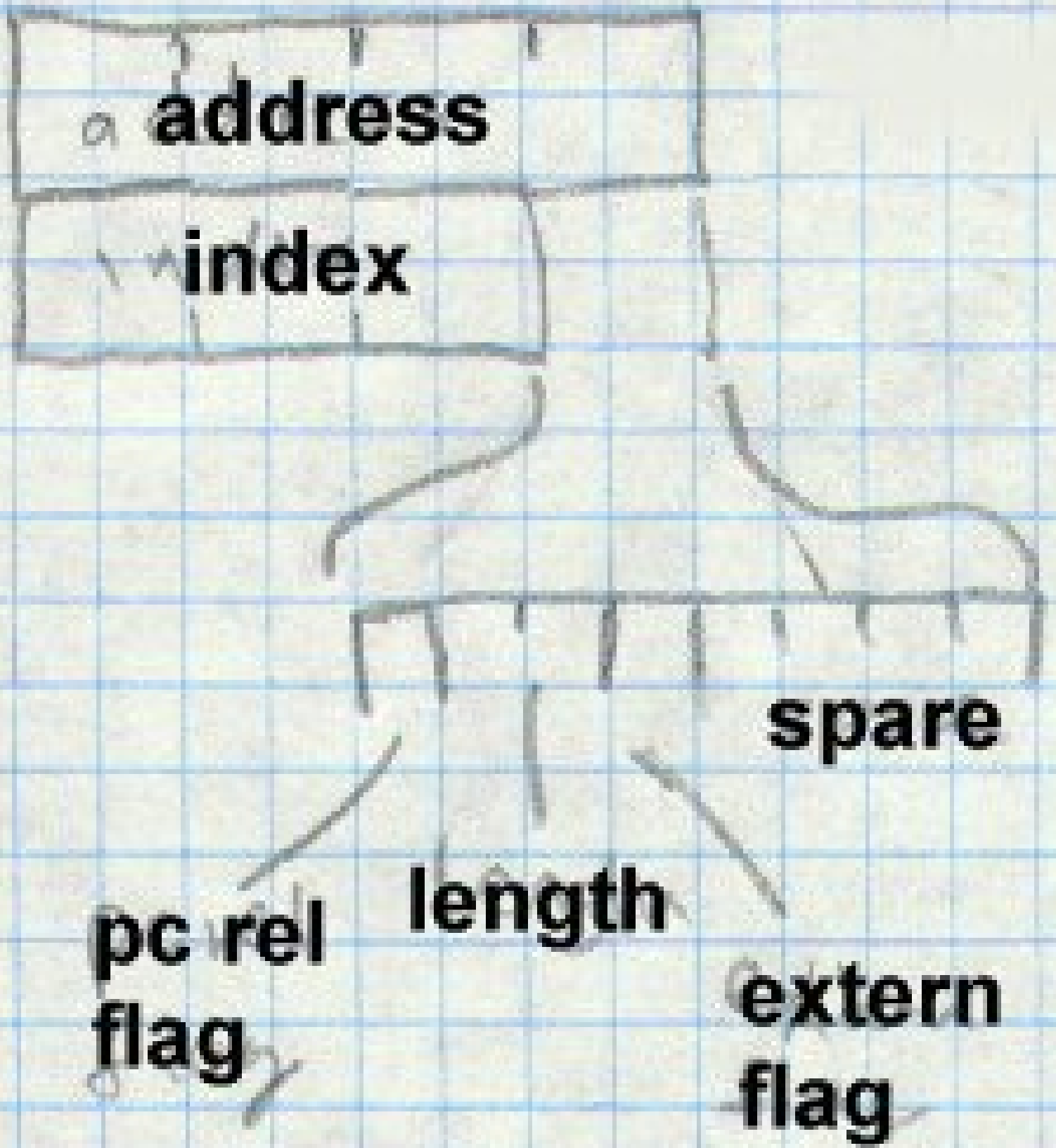

a.out
file
hdr
text
data
text size
data size
process
text
segment
data
bss
heap
bss size
from a.out
header
stack

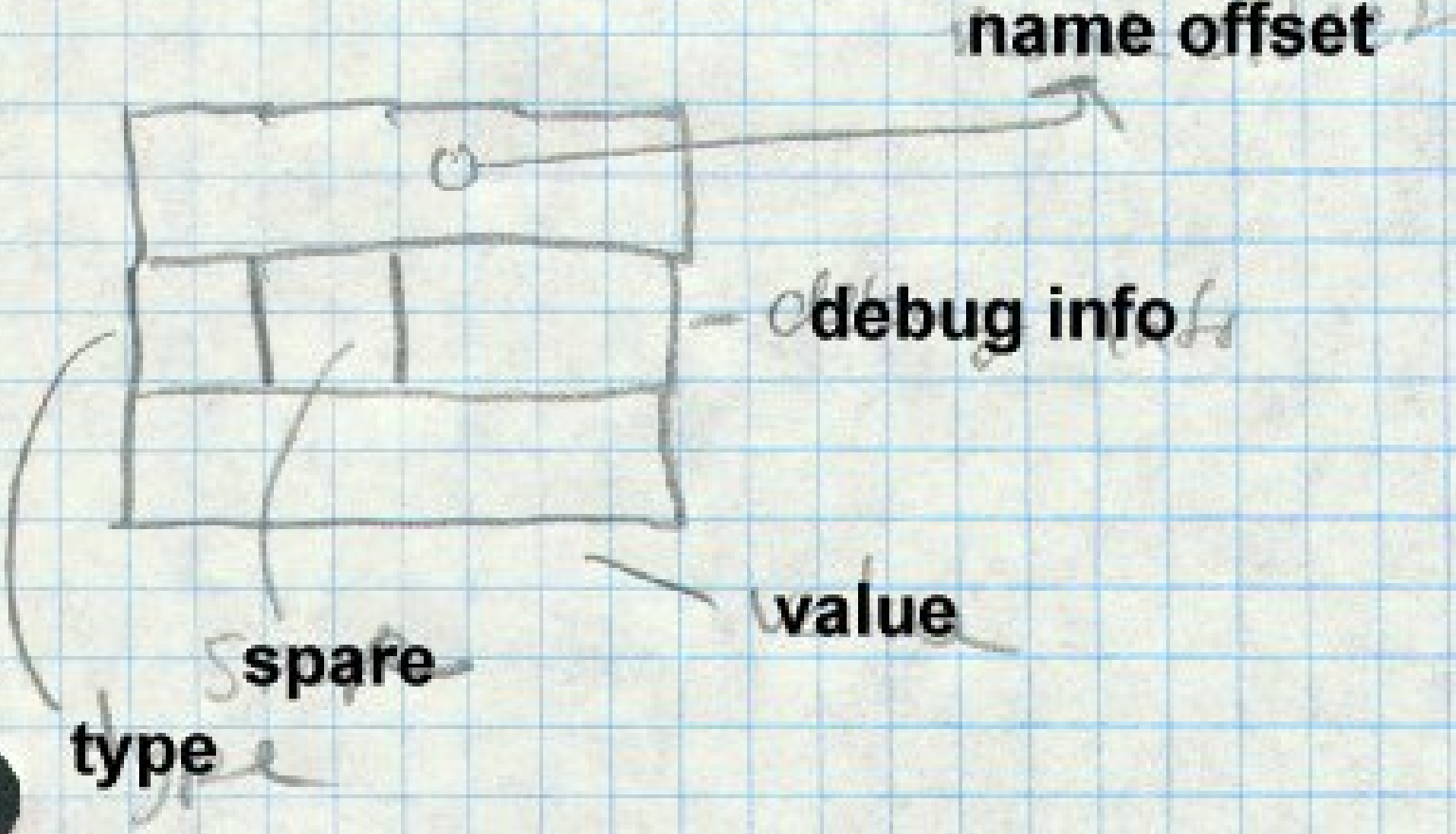

pagable
a.out file
header page
(not mapped)
process
text
pages
read only
mapped pages
address 0
text
segment
(read-only)
data
pages
copy on write
mapped pages
data
bss
heap
read
write
stack
read
write

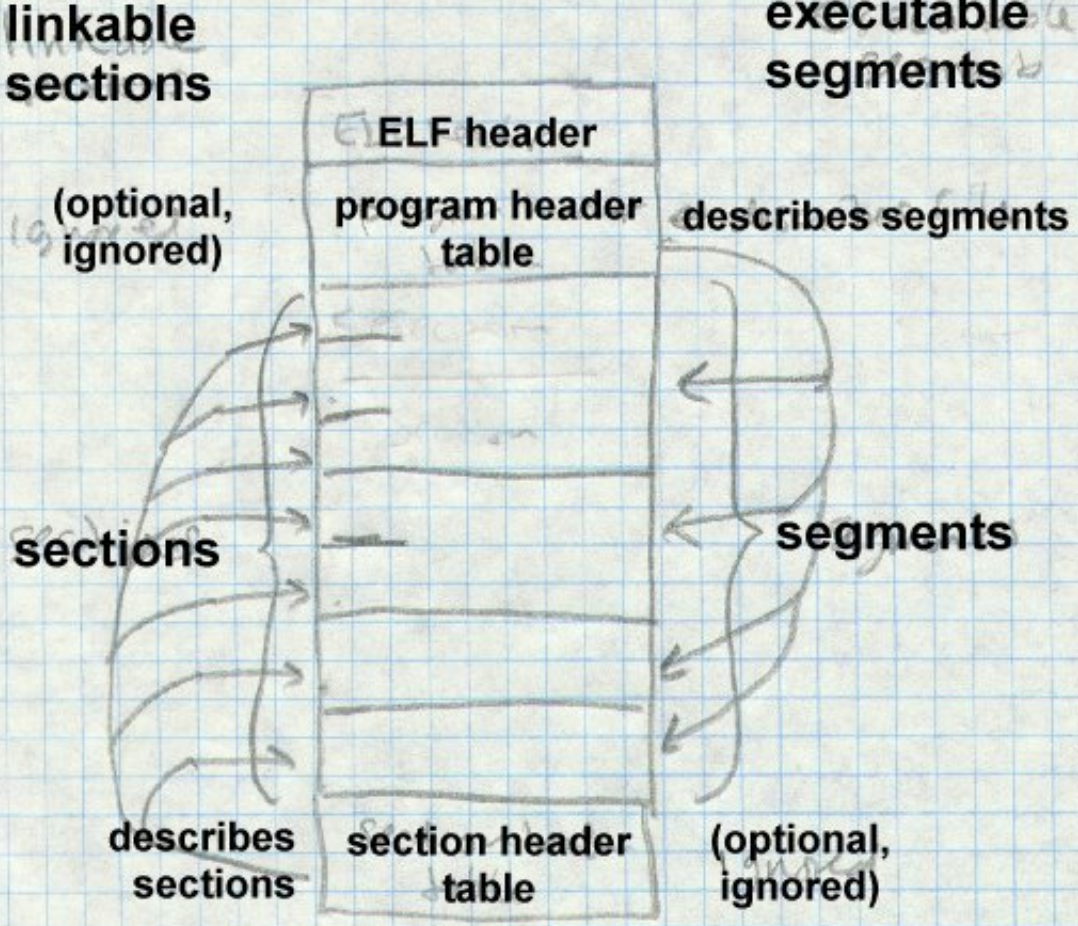


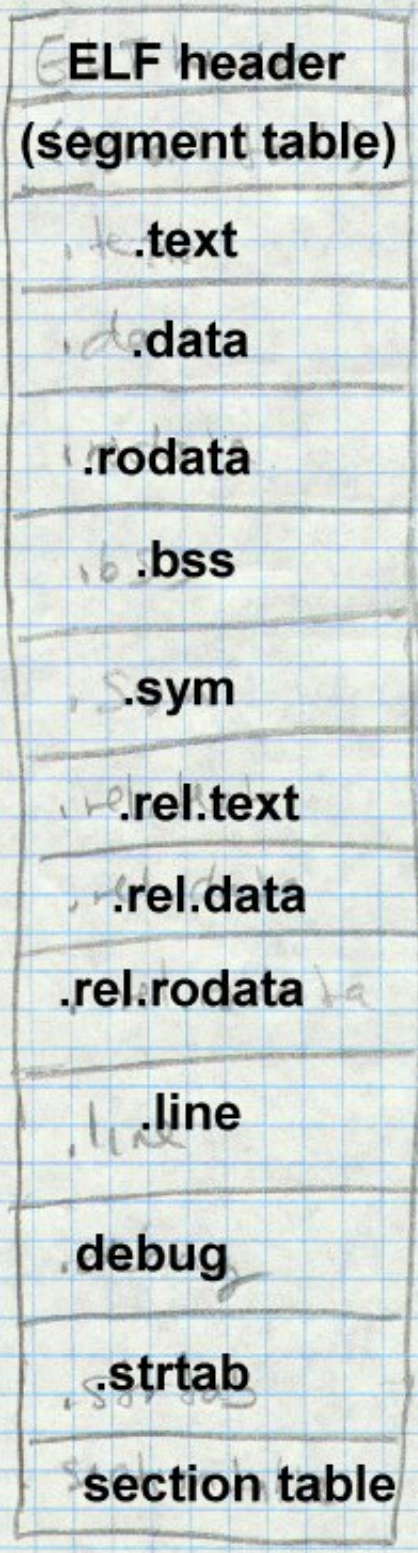
Note: this figure is supposed to be almost the same as 3-4, so I've put the different stuff in red.

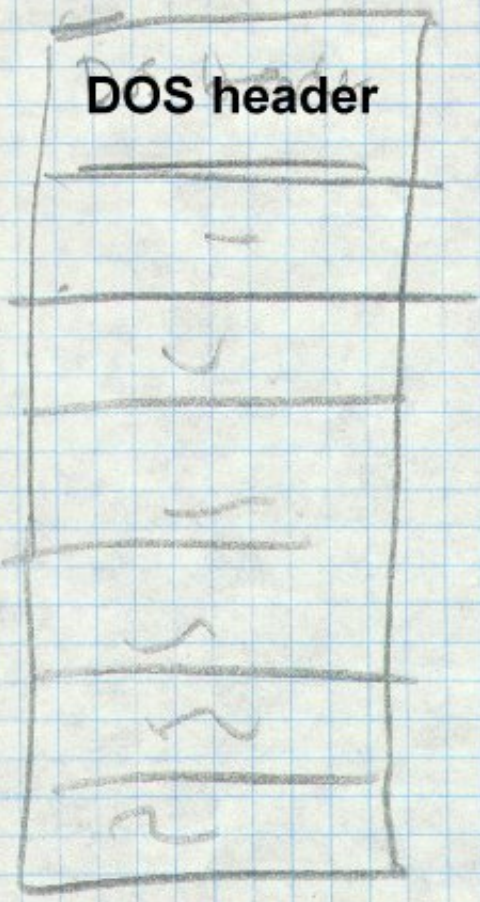

a.out header
text
data
text
reloc
data
reloc
symbol
table
string
table


address
index
spare
pc rel flag
length
extern flag


name offset
debug info
value
spare
type


linkable sections
executable segments
ELF header
(optional, ignored)
program header table
describes segments
sections
segments
describes sections
section header table
(optional, ignored)

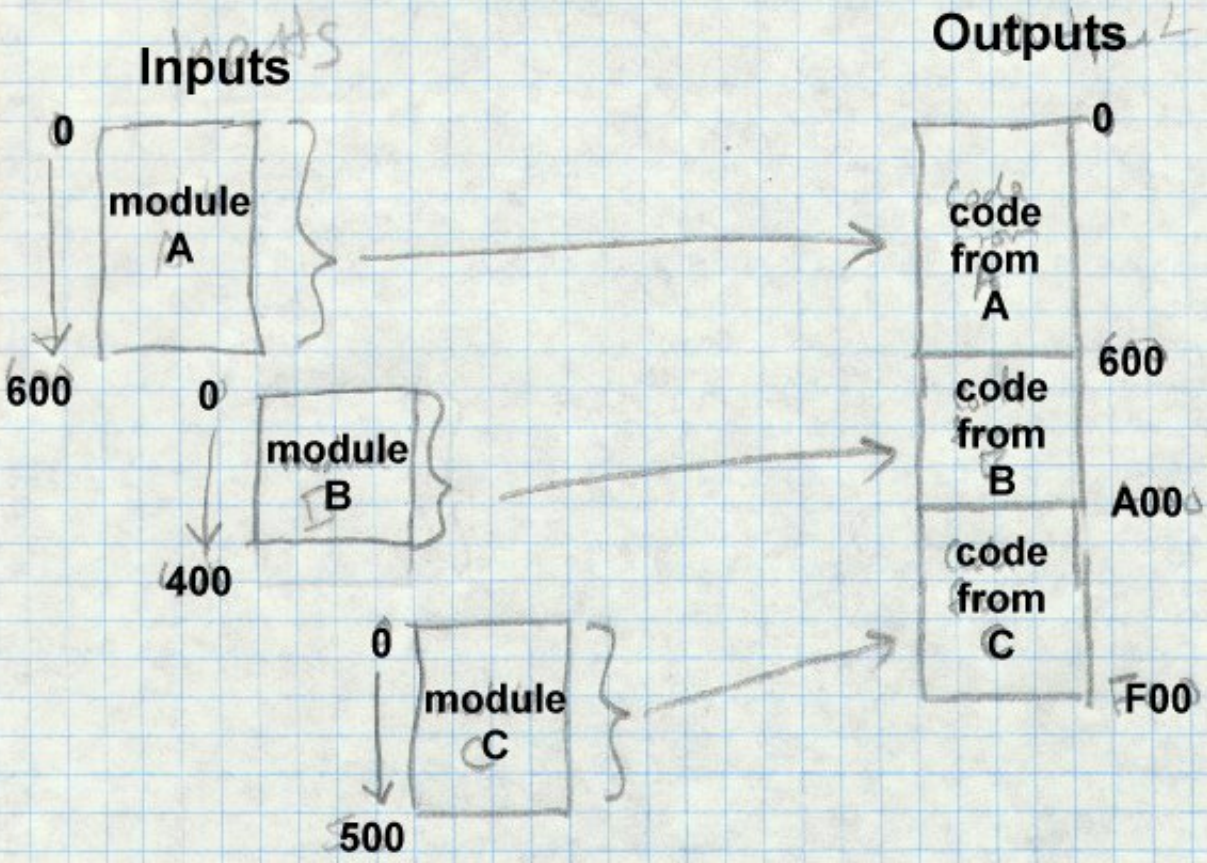

ELF header
(not considered sections)
(segment table)
.text
.data
.rodata
.bss
.sym
.rel.text
.rel.data
.rel.rodata
.line
.debug
.strtab
section table
(not considered a section)

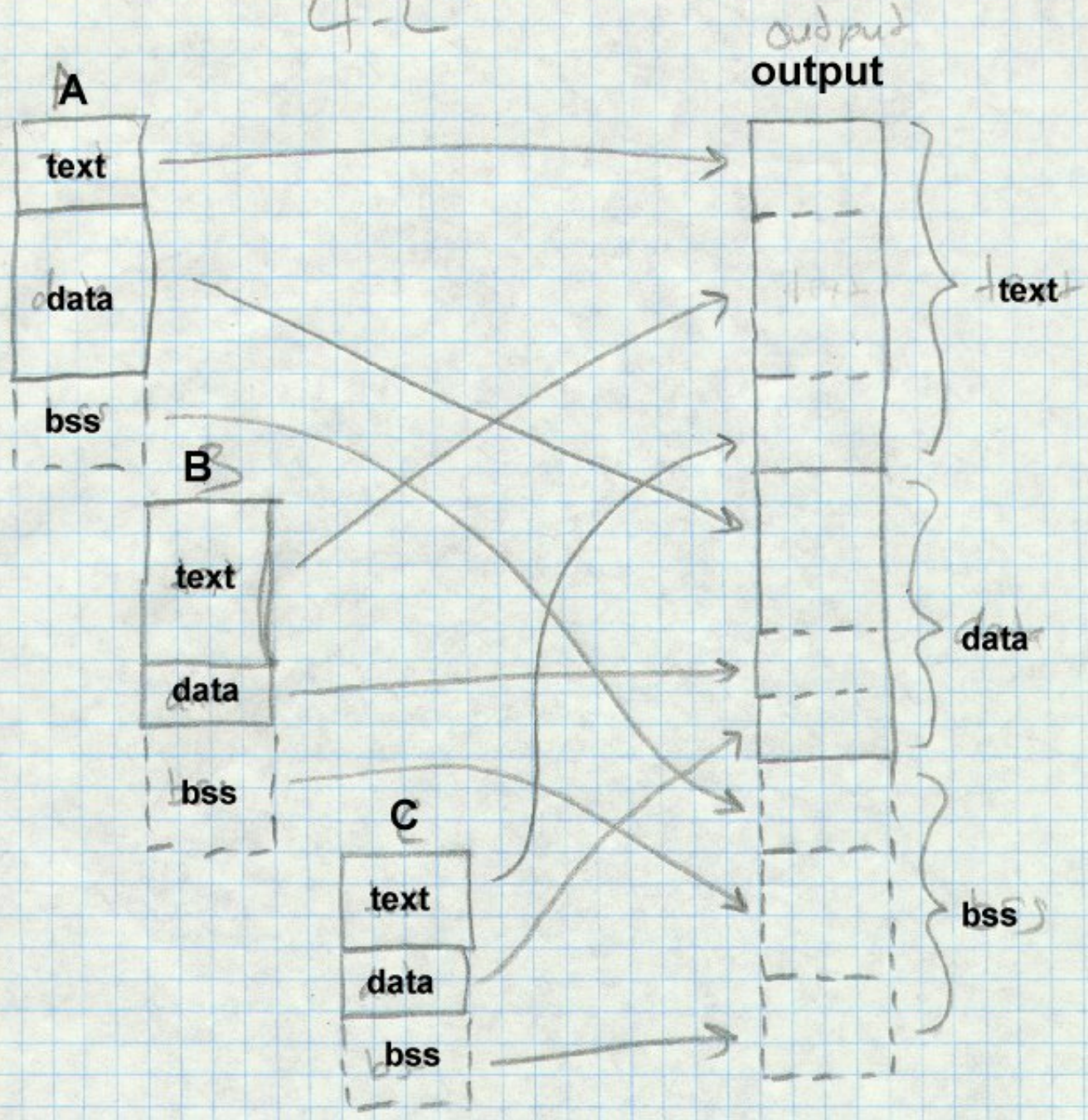


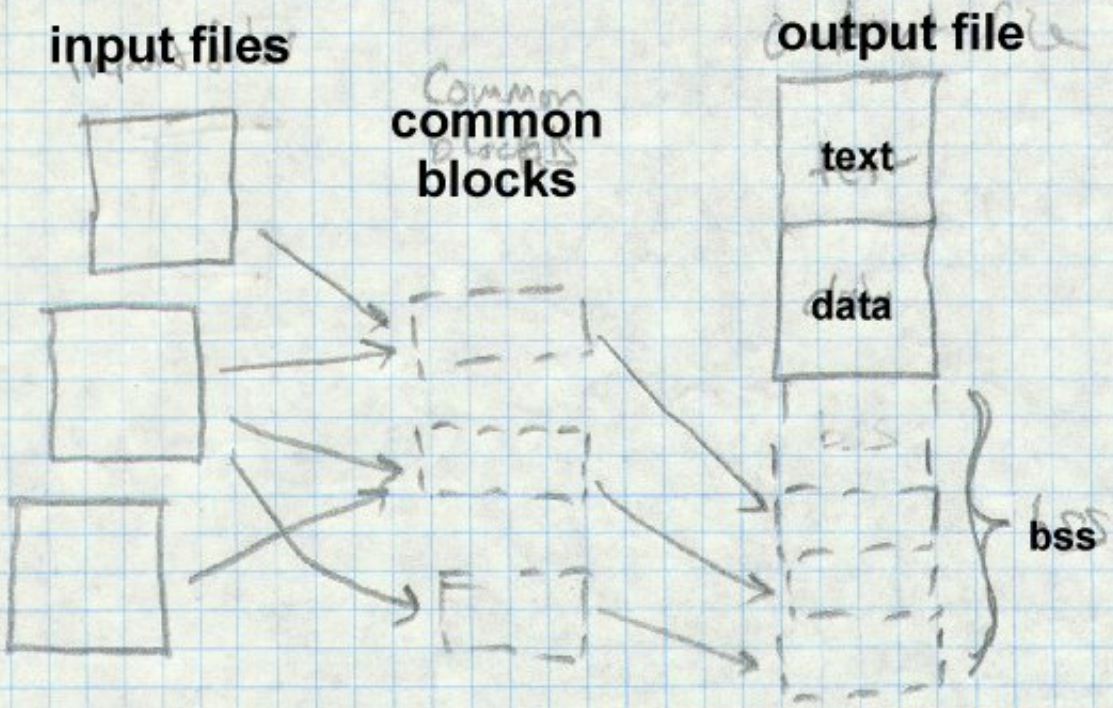
See chapter file for box captions

See callouts in chapter for the captions for all these boxes.

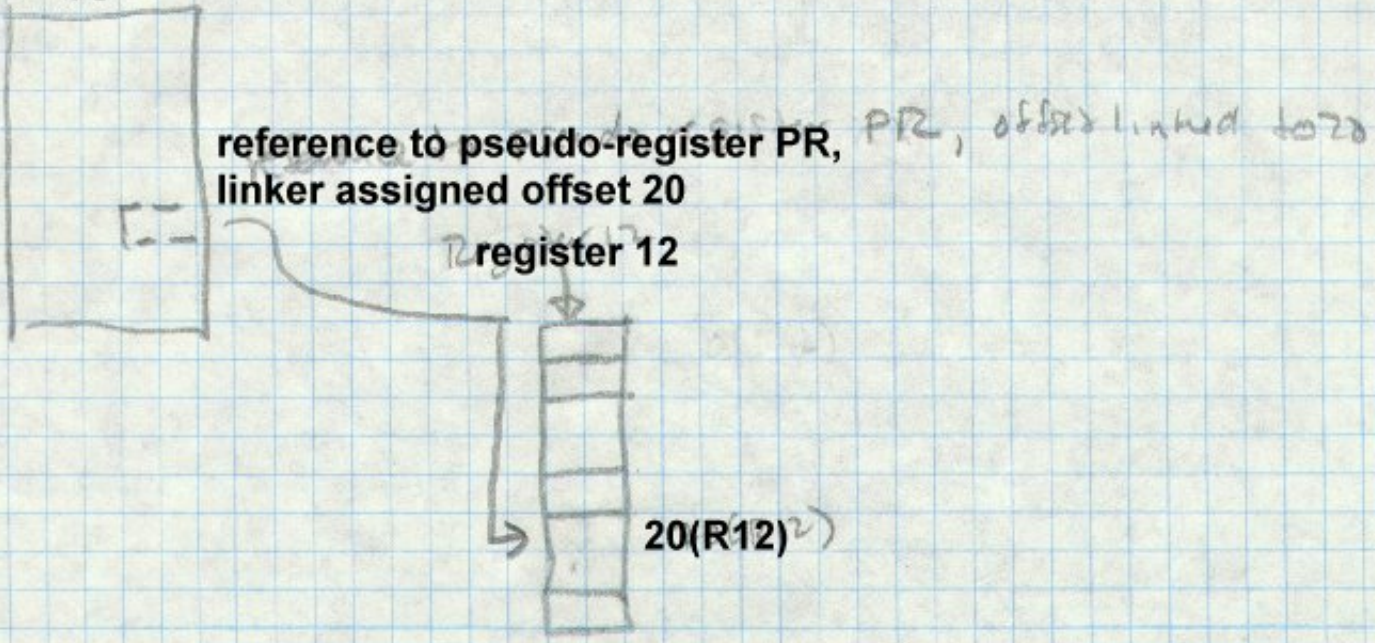

record type
length
data
checksum

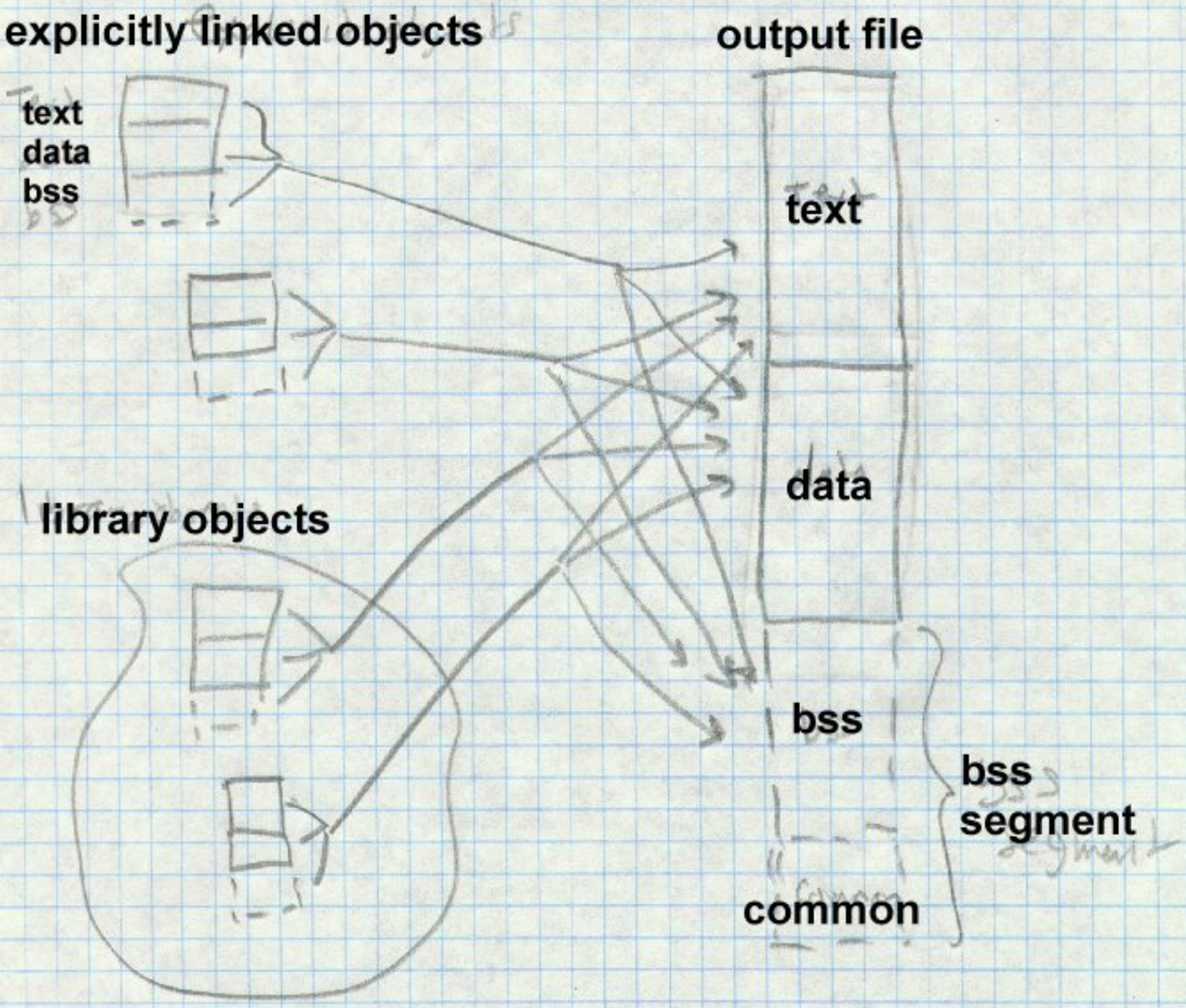

Inputs
Outputs
0
module A
600
0
module B
400
0
module C
500
0
code from A
600
code from B
A00
code from C
F00

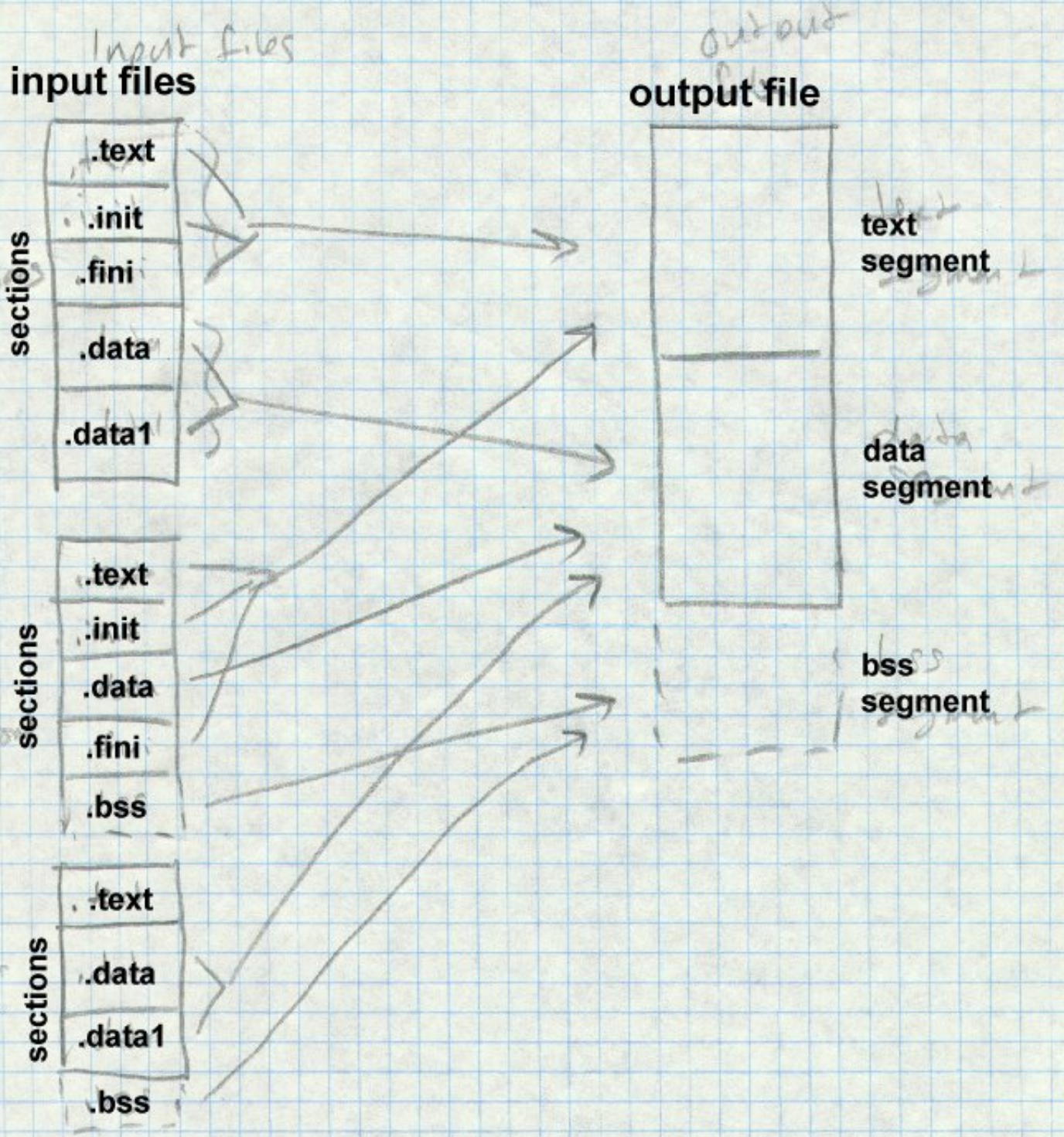

4-2
output
A
text
data
bss
B
text
data
bss
C
text
data
bss
output
text
data
bss

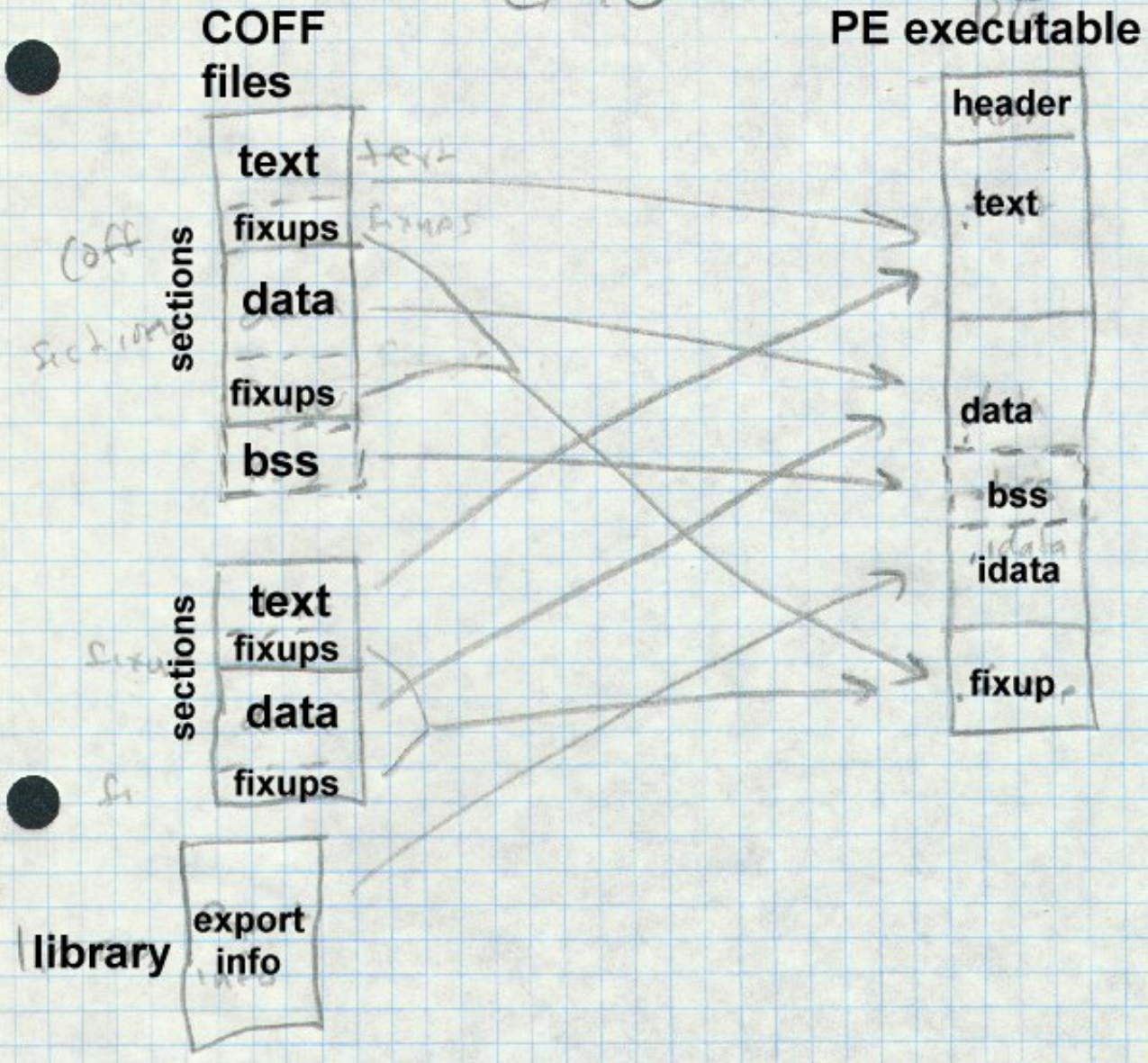

input files
common blocks
output file
text
data
bss

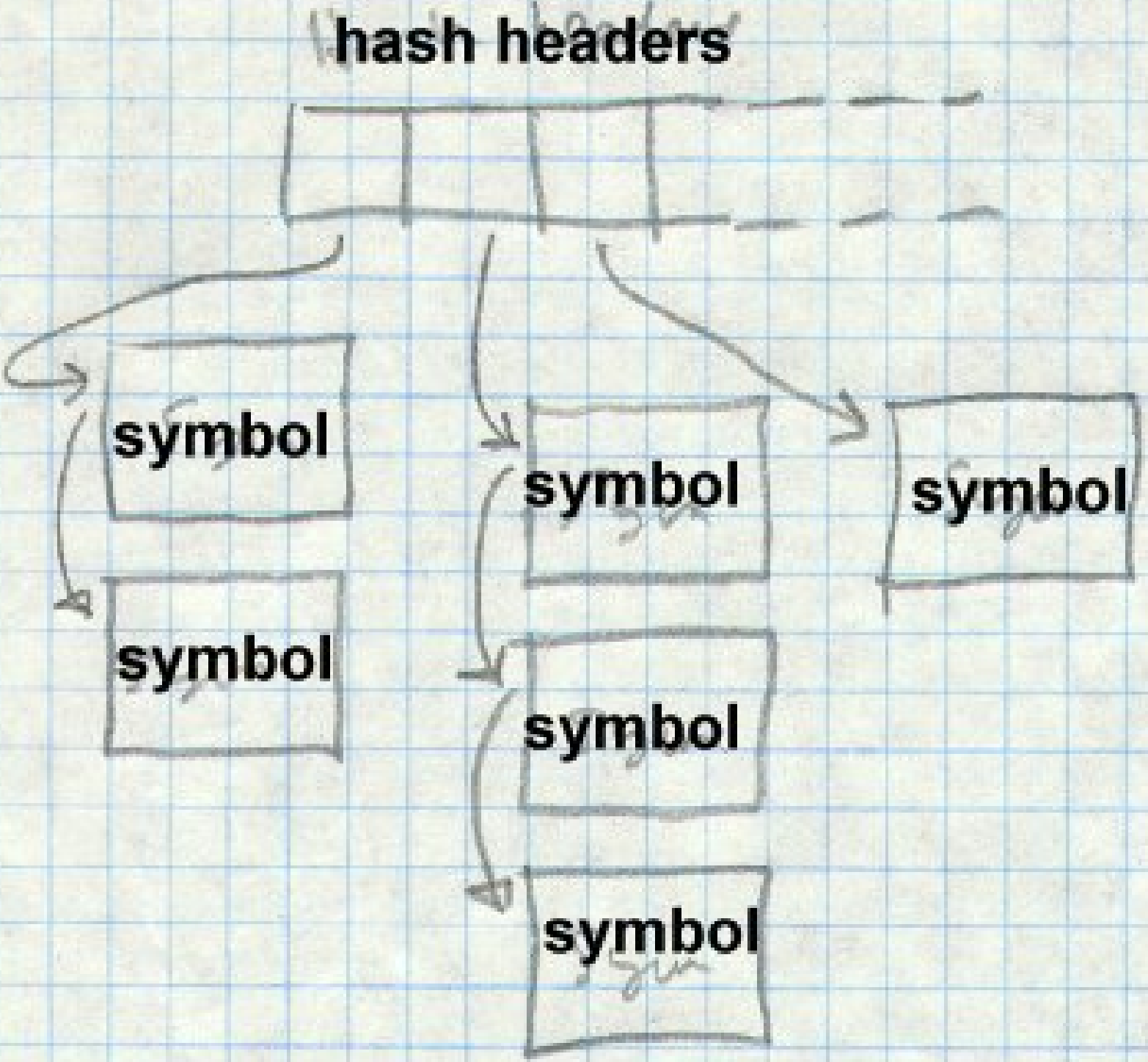
object file

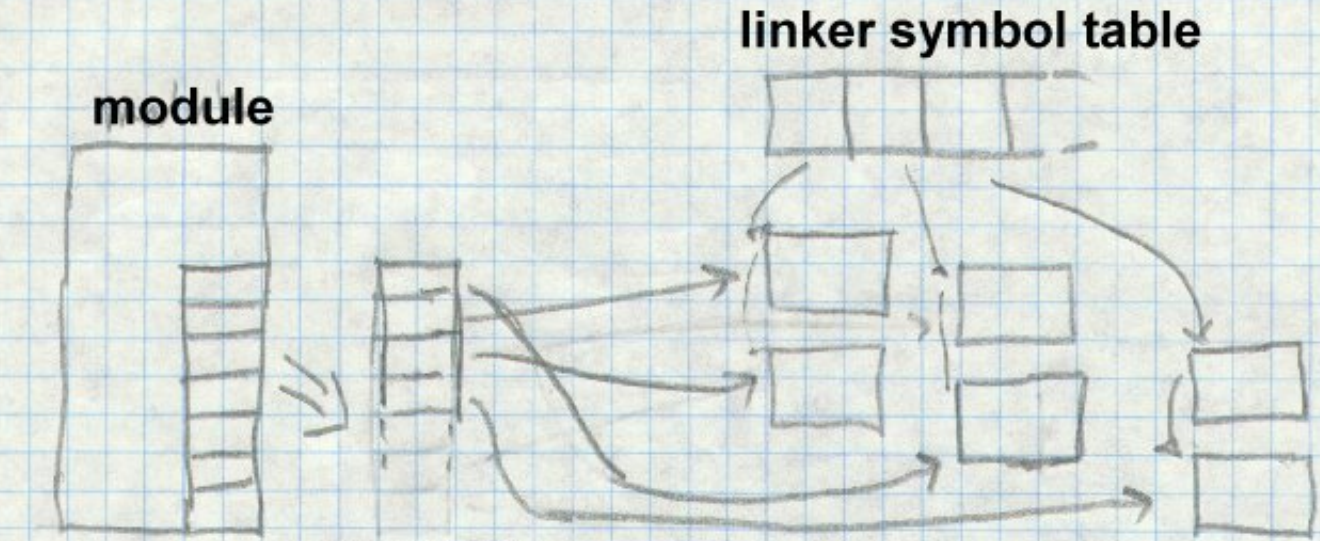

reference to pseudo-register PR,
linker assigned offset 20
register 12
20(R12)


explicitly linked objects
output file
text
data
bss
text
data
library objects
bss
bss
segment
common

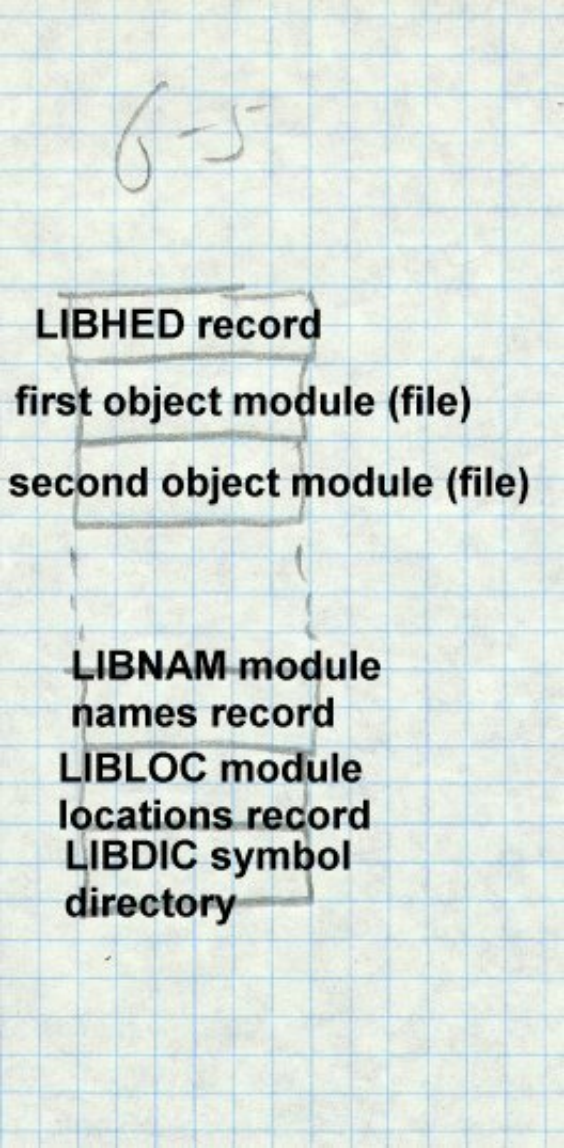

input files
output file
sections
.text
.init
.fini
.data
.data1
sections
.text
.init
.data
.fini
.bss
sections
.text
.data
.data1
.bss
text segment
data segment
bss segment


COFF files
sections
text
fixups
data
fixups
bss
sections
text
fixups
data
fixups
library
export info
PE executable
header
text
data
bss
idata
fixup


hash headers
symbol
symbol
symbol
symbol
symbol
symbol

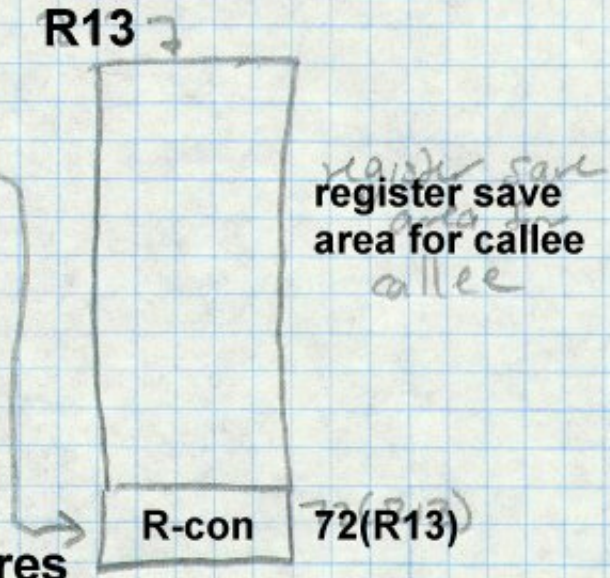

linker symbol table
module
symbol table, indexed by entry number
turned into list of pointers into symbol table

6-5


LIBHED record
first object module (file)
second object module (file)
LIBNAM module names record
LIBLOC module locations record
LIBDIC symbol directory

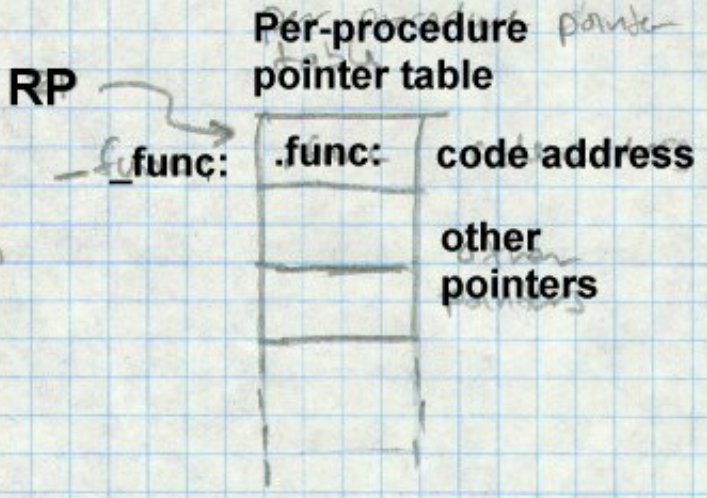
Caller:
- copy R-con into save area
- load V-con into R15
- Call via R15

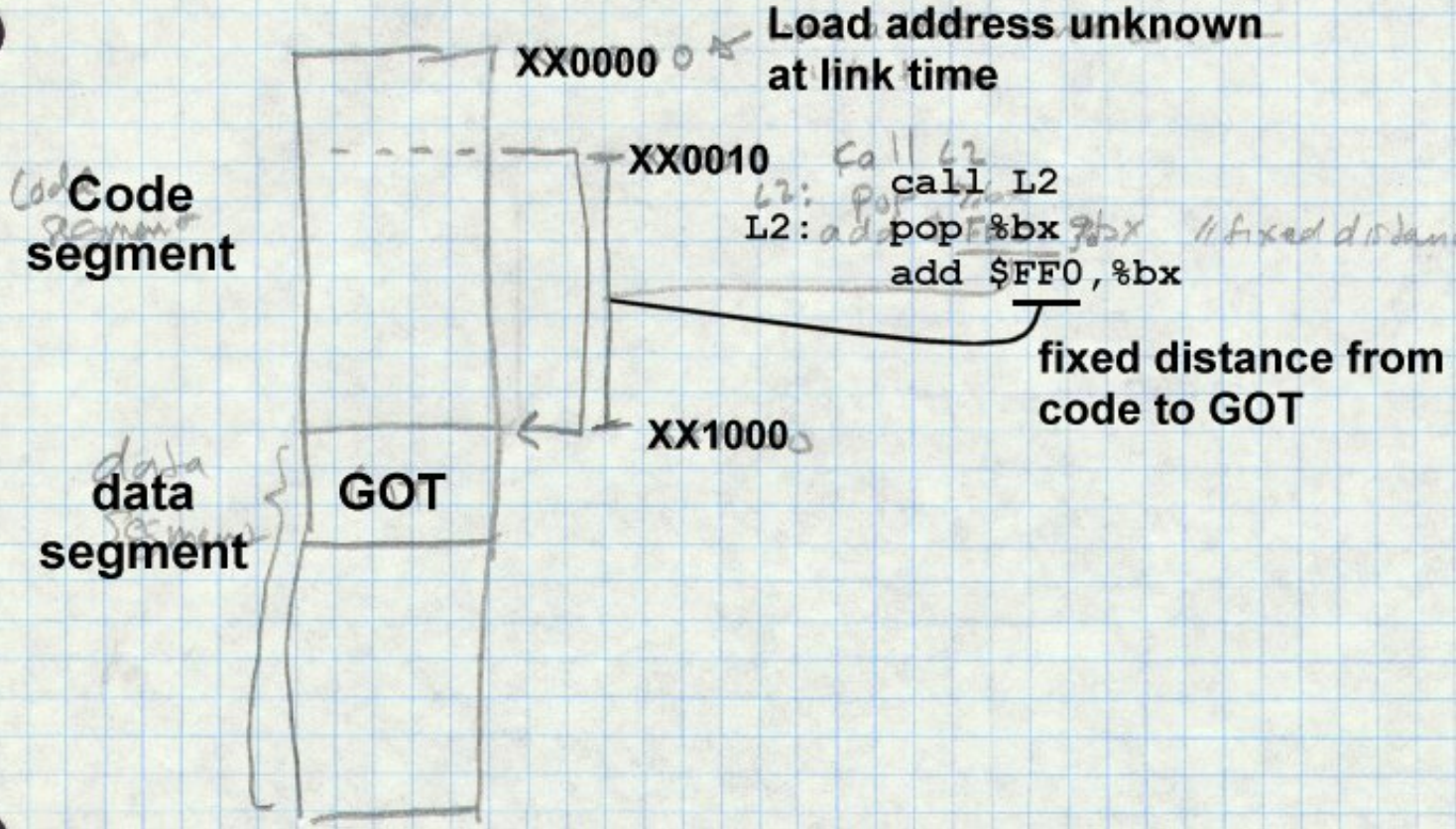
Callee:
- load R-con from save area
- addresses of sub-procedures in data area

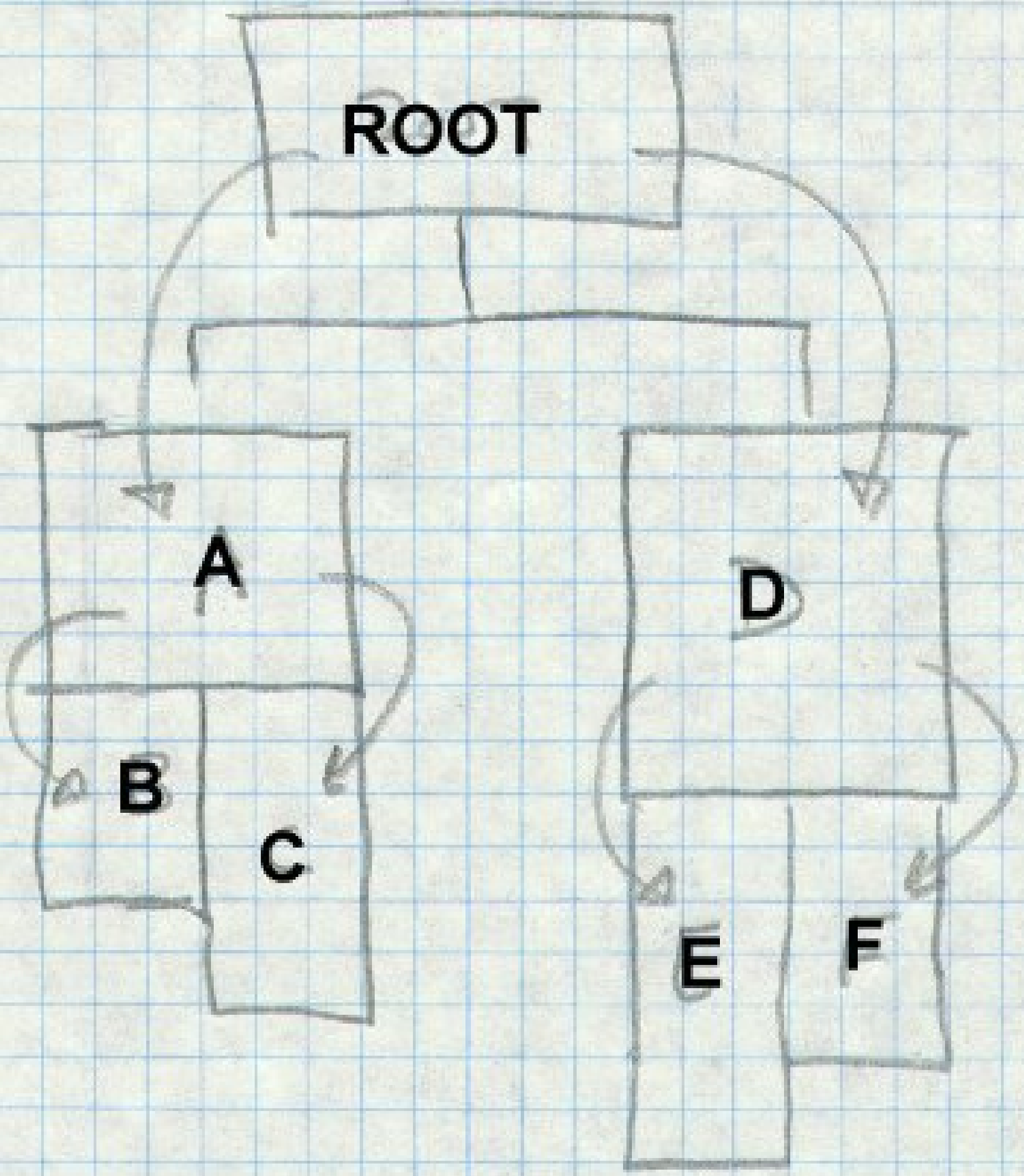


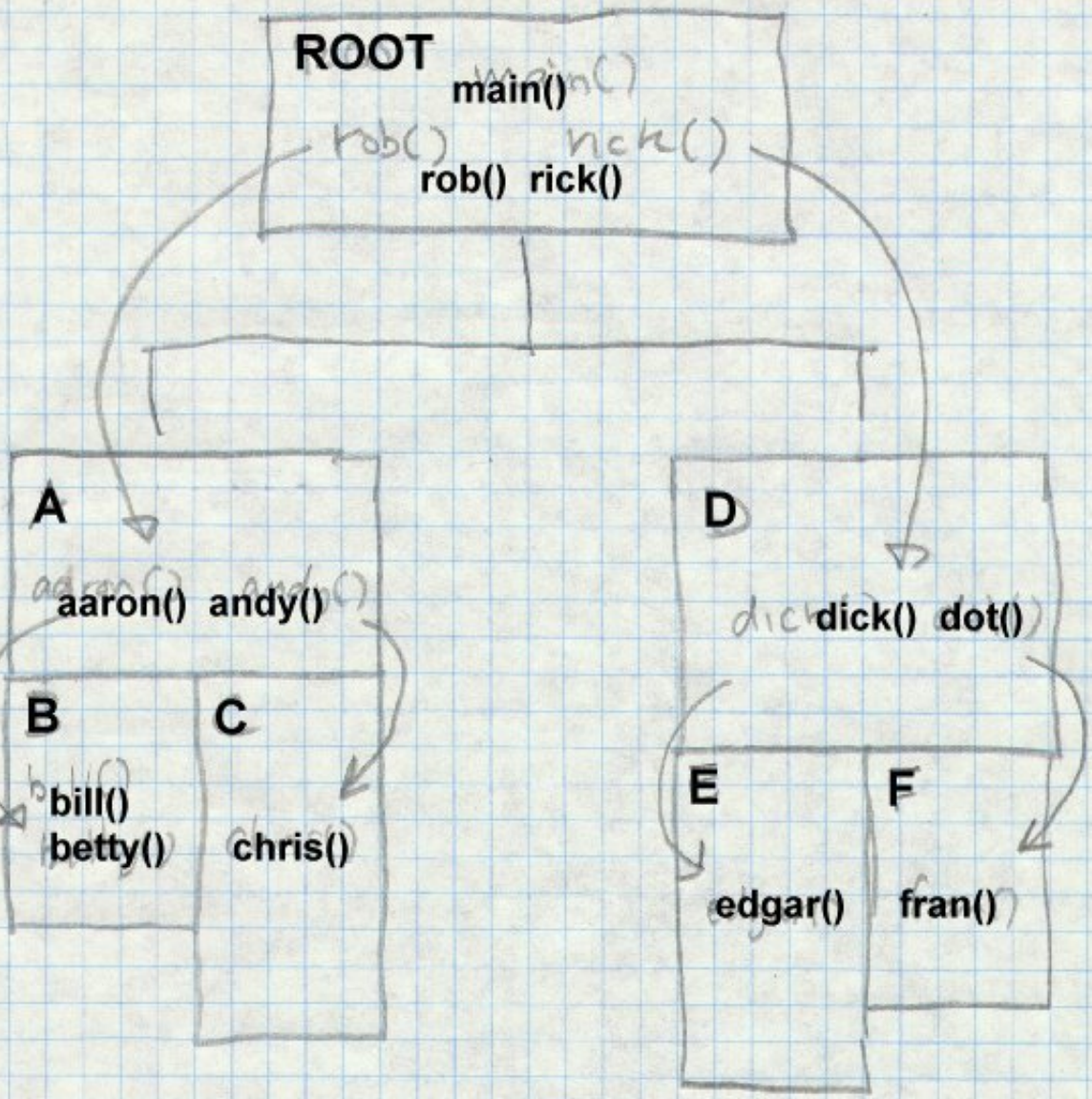
Caller:
- Load pointer table address into RP
- Load code address from 0(RP) into RC
- Call via RC

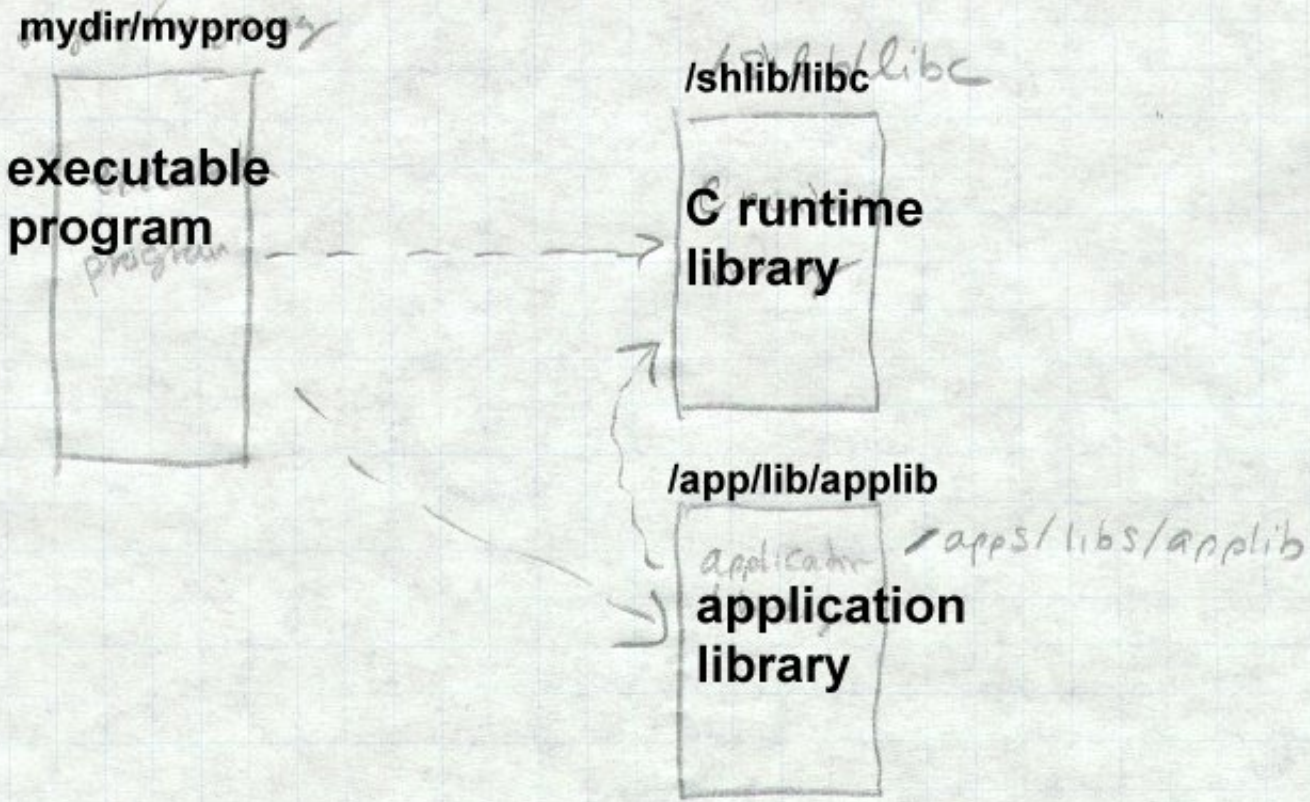
Callee:
- RP points to pointer table
- Table has addresses of pointer tables for sub-procedures

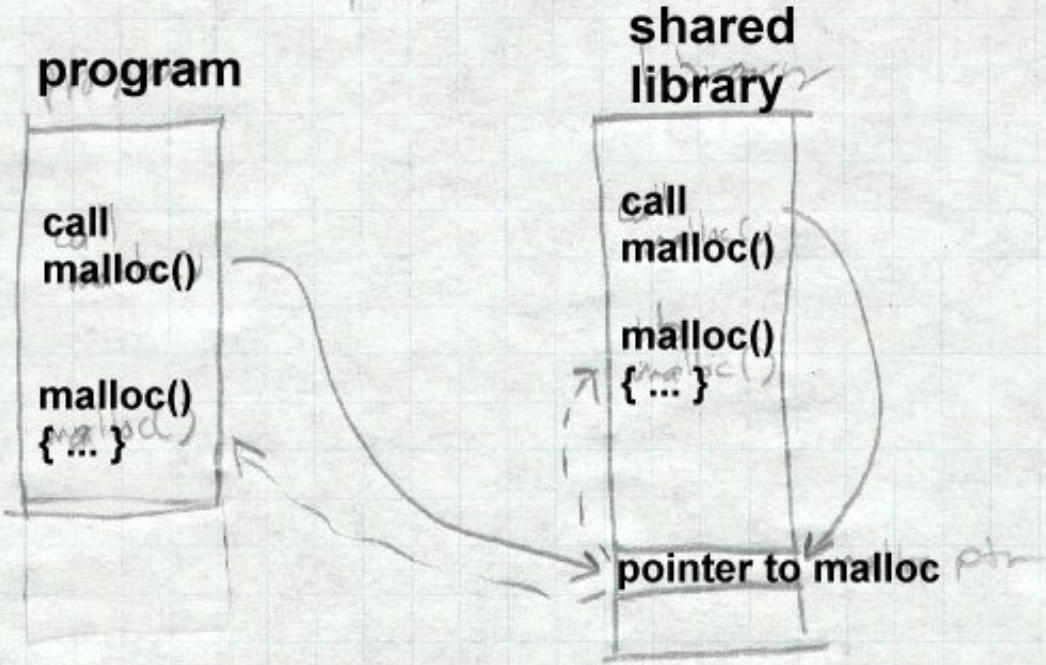


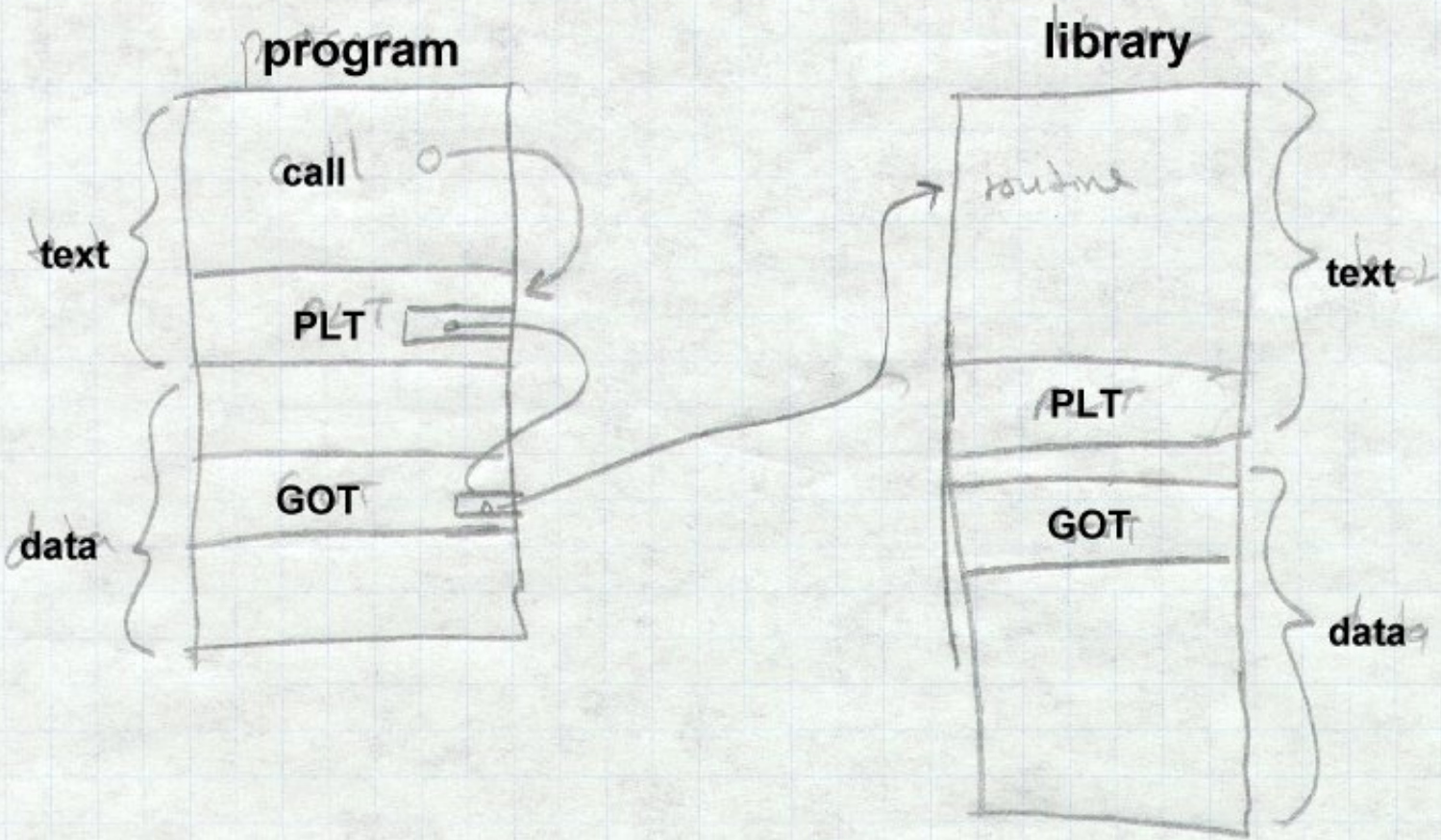

Load address unknown
at link time
XX0000
XX0010
call L2
L2: pop %bx
add $FF0,%bx
Code
segment
fixed distance from
code to GOT
XX1000
data
segment
GOT

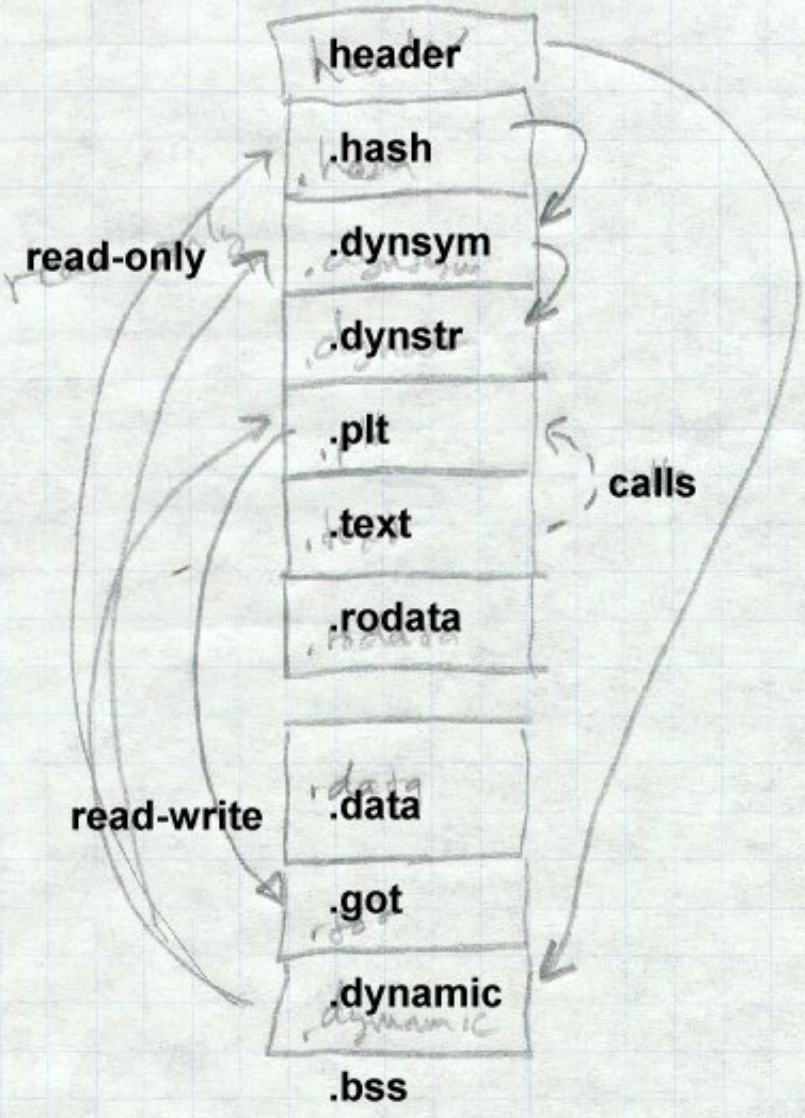

ROOT
A
B
C
D
E
F

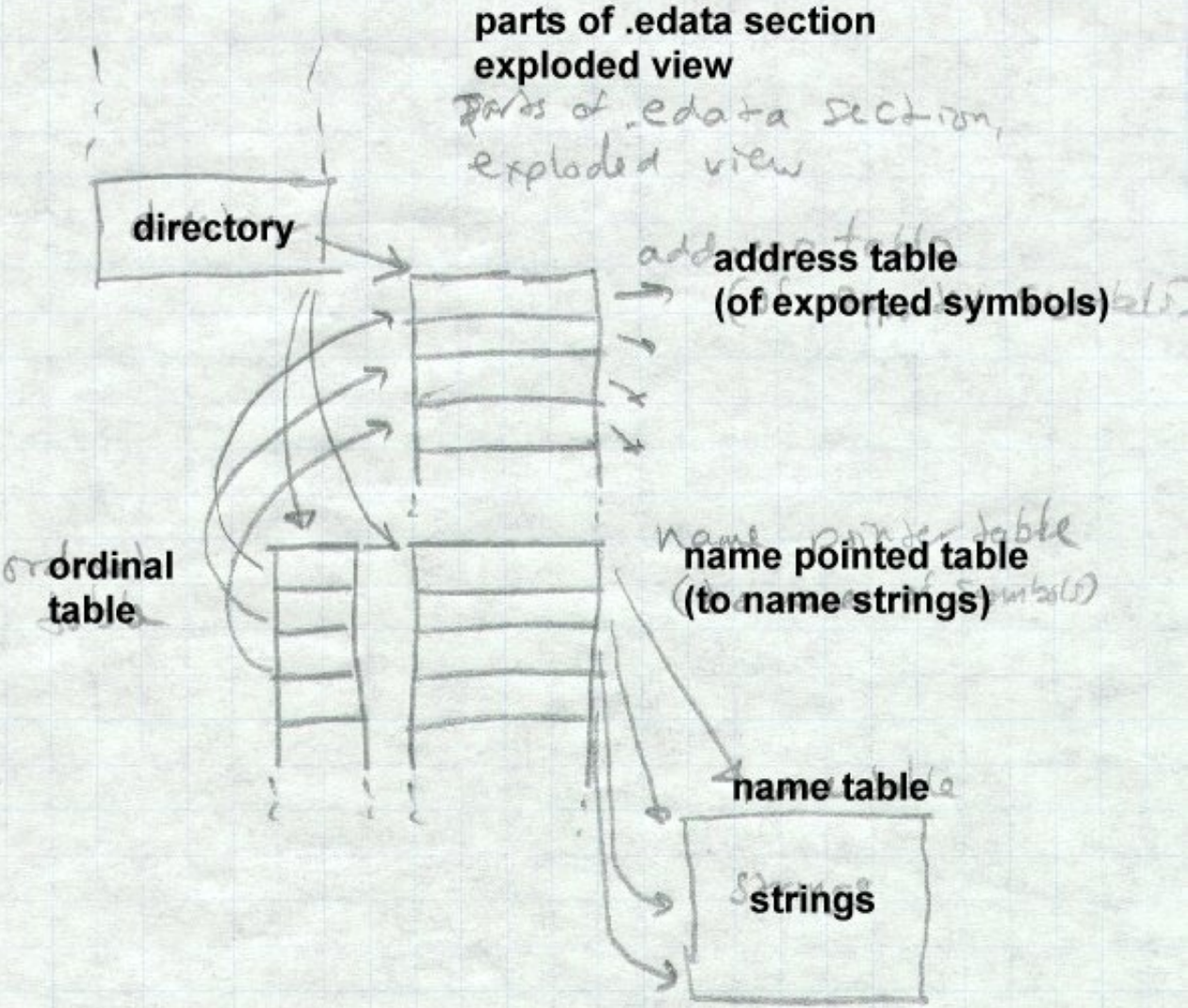

ROOT
main()
rob() rick()
A
aaron() andy()
B
bill()
betty()
C
chris()
D
dick() dot()
E
edgar()
F
fran()

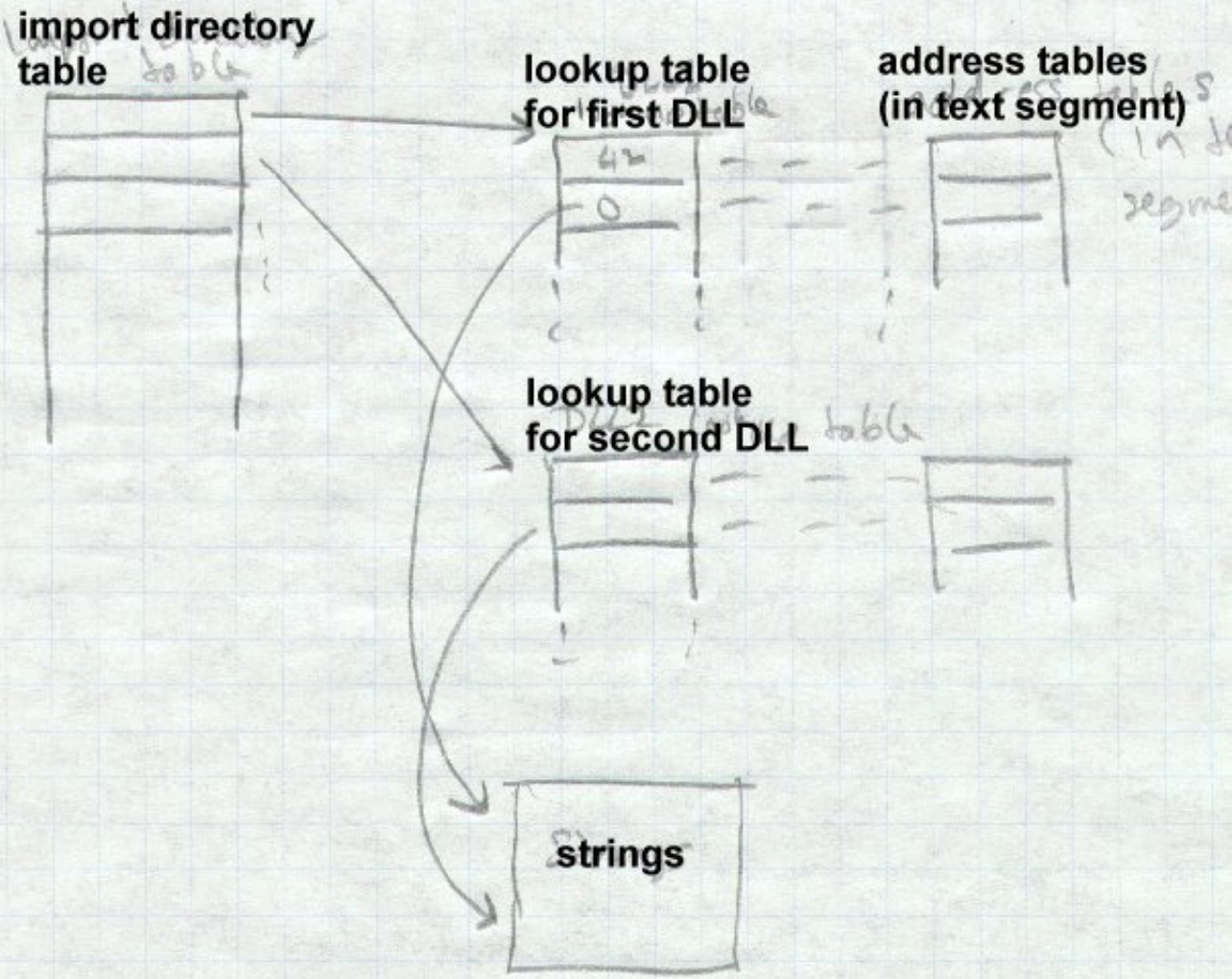

mydir/myprog
executable program
/shlib/libc
C runtime library
/app/lib/applib
application library

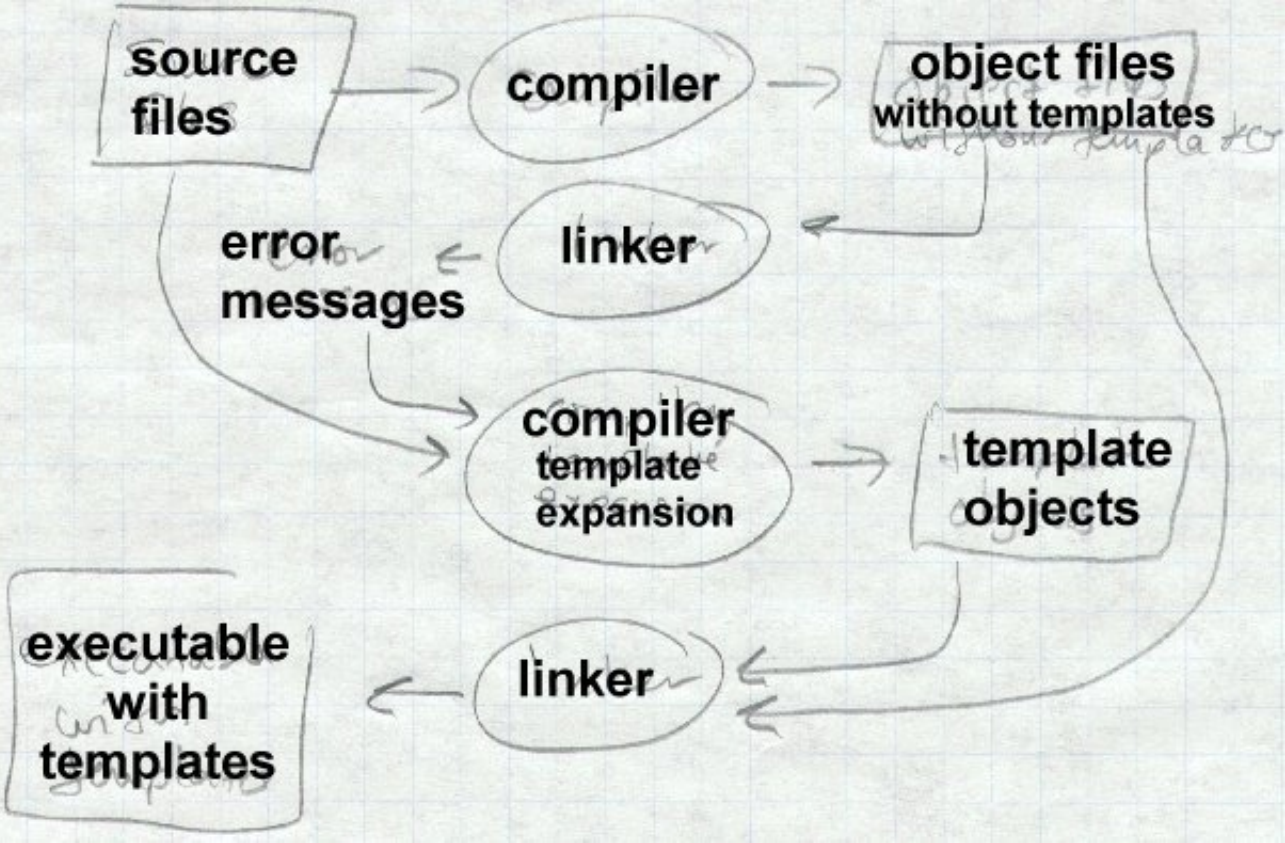

program
call
malloc()
malloc()
{ ... }
shared
library
call
malloc()
malloc()
{ ... }
pointer to malloc

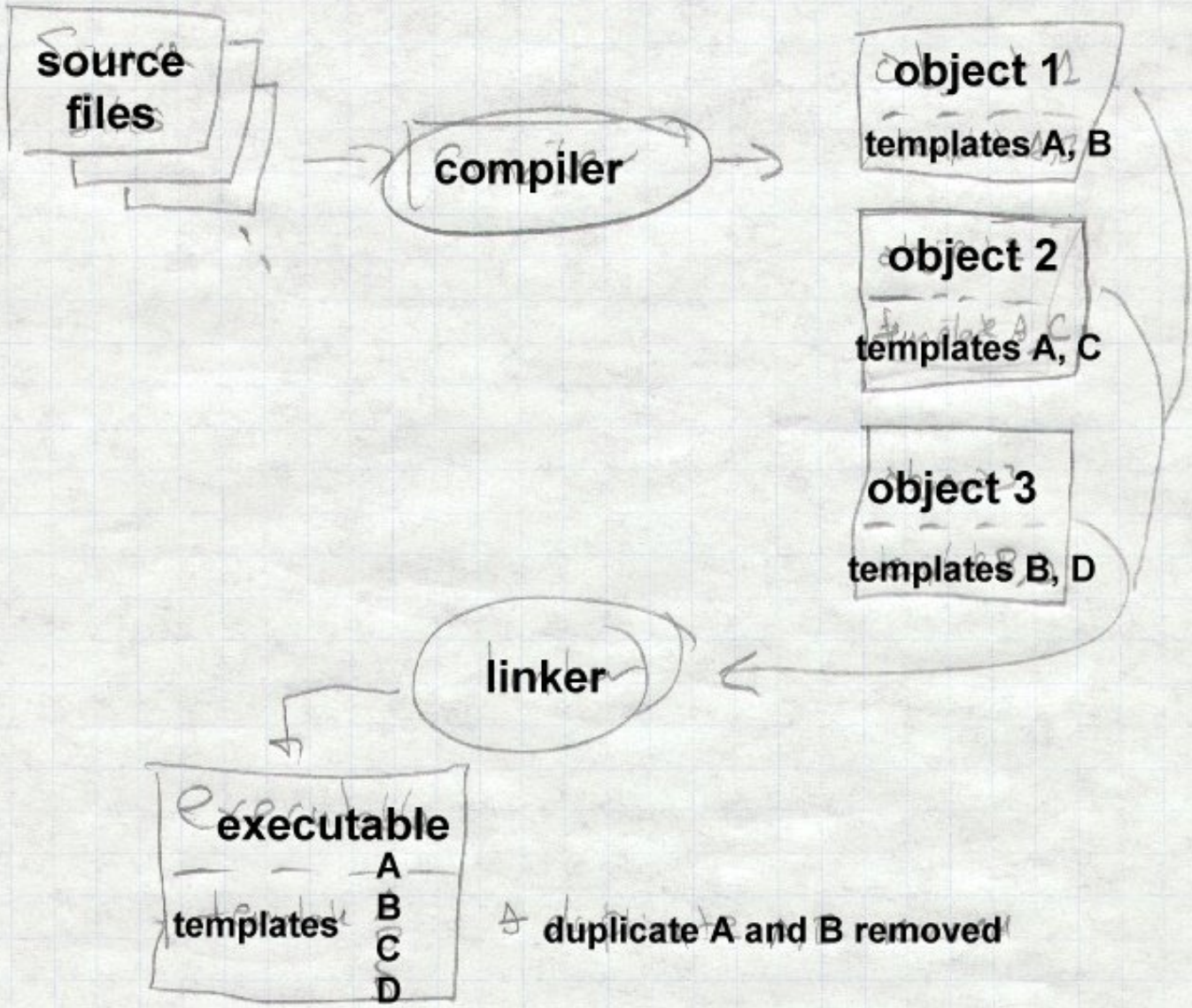

program
call
text
PLT
GOT
data
library
routine
text
PLT
GOT
data

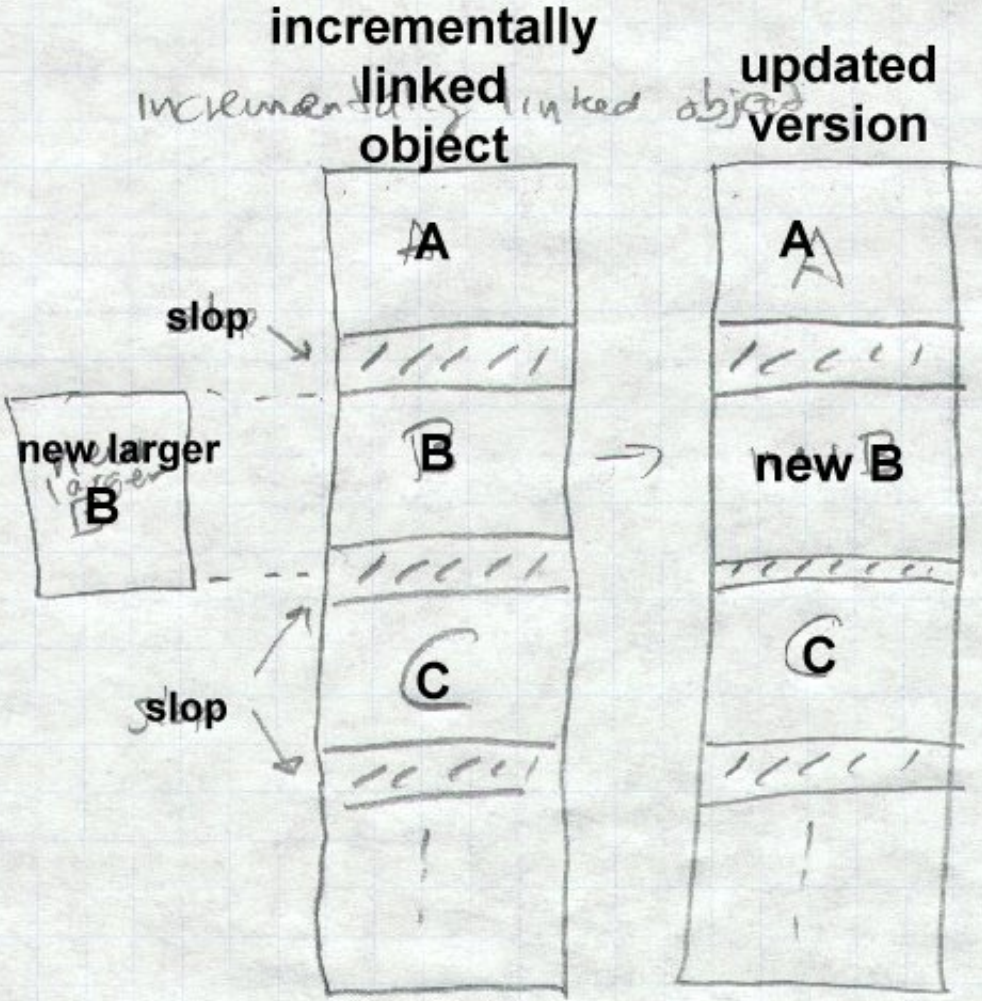

header
.hash
.dynsym
.dynstr
.plt
.text
.rodata
.data
.got
.dynamic
.bss
read-only
read-write
calls

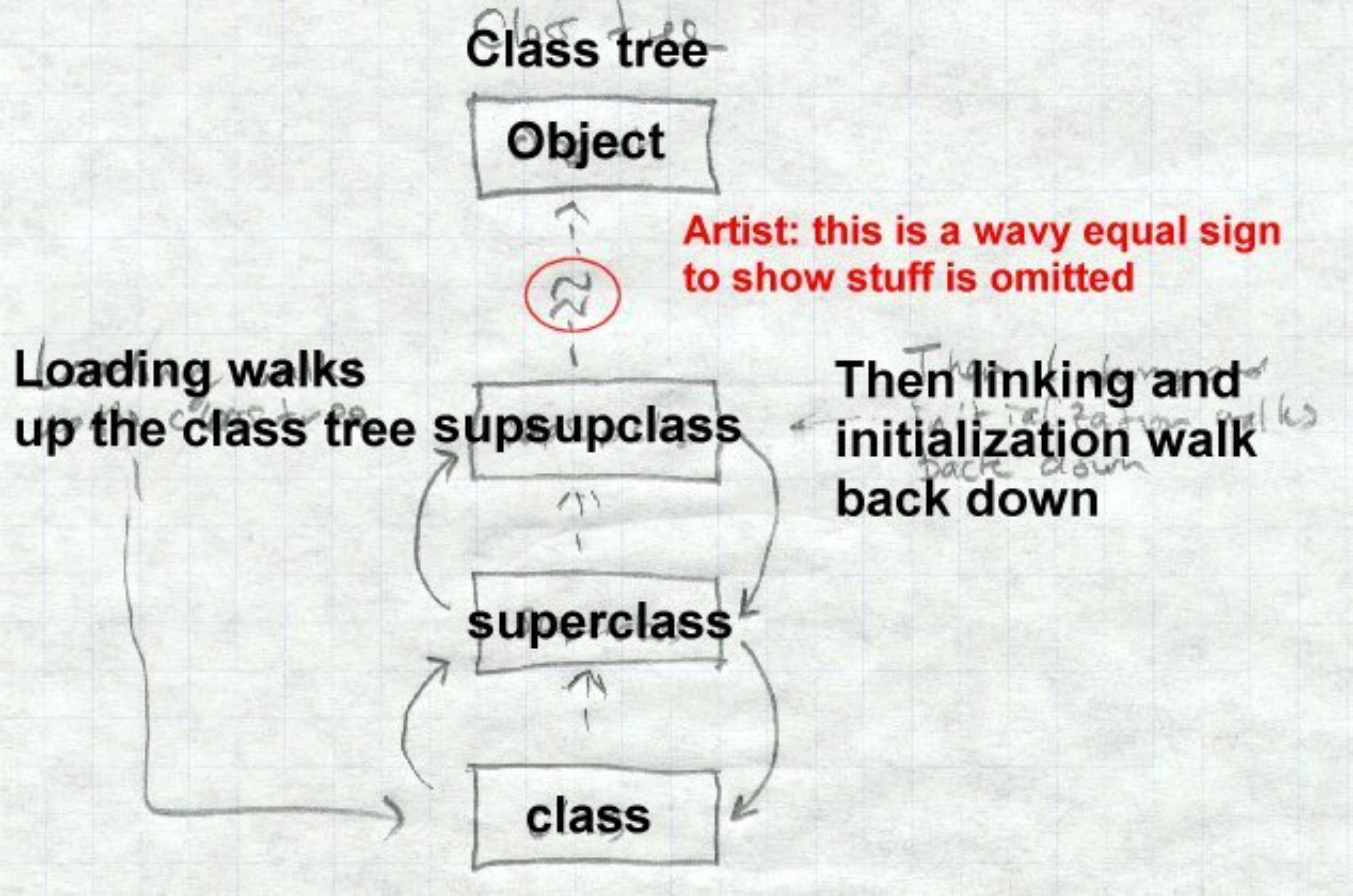

parts of .edata section
exploded view
directory
address table
(of exported symbols)
ordinal
table
name pointed table
(to name strings)
name table
strings


import directory table
lookup table for first DLL
address tables (in text segment)
lookup table for second DLL
strings


source files
compiler
object files without templates
linker
error messages
compiler template expansion
template objects
linker
executable with templates


source files
compiler
object 1
templates A, B
object 2
templates A, C
object 3
templates B, D
linker
executable
templates
A
B
C
D
duplicate A and B removed


incrementally linked object
updated version
A
slop
B
new larger B
C
slop
A
new B
C
B relinked in place, replaces old B and part of slop


Class tree
Object
Artist: this is a wavy equal sign to show stuff is omitted
Loading walks up the class tree
supsupclass
Then linking and initialization walk back down
superclass
class