Final Review

ICS312

Machine-Level and Systems Programming

Henri Casanova (henric@hawaii.edu)

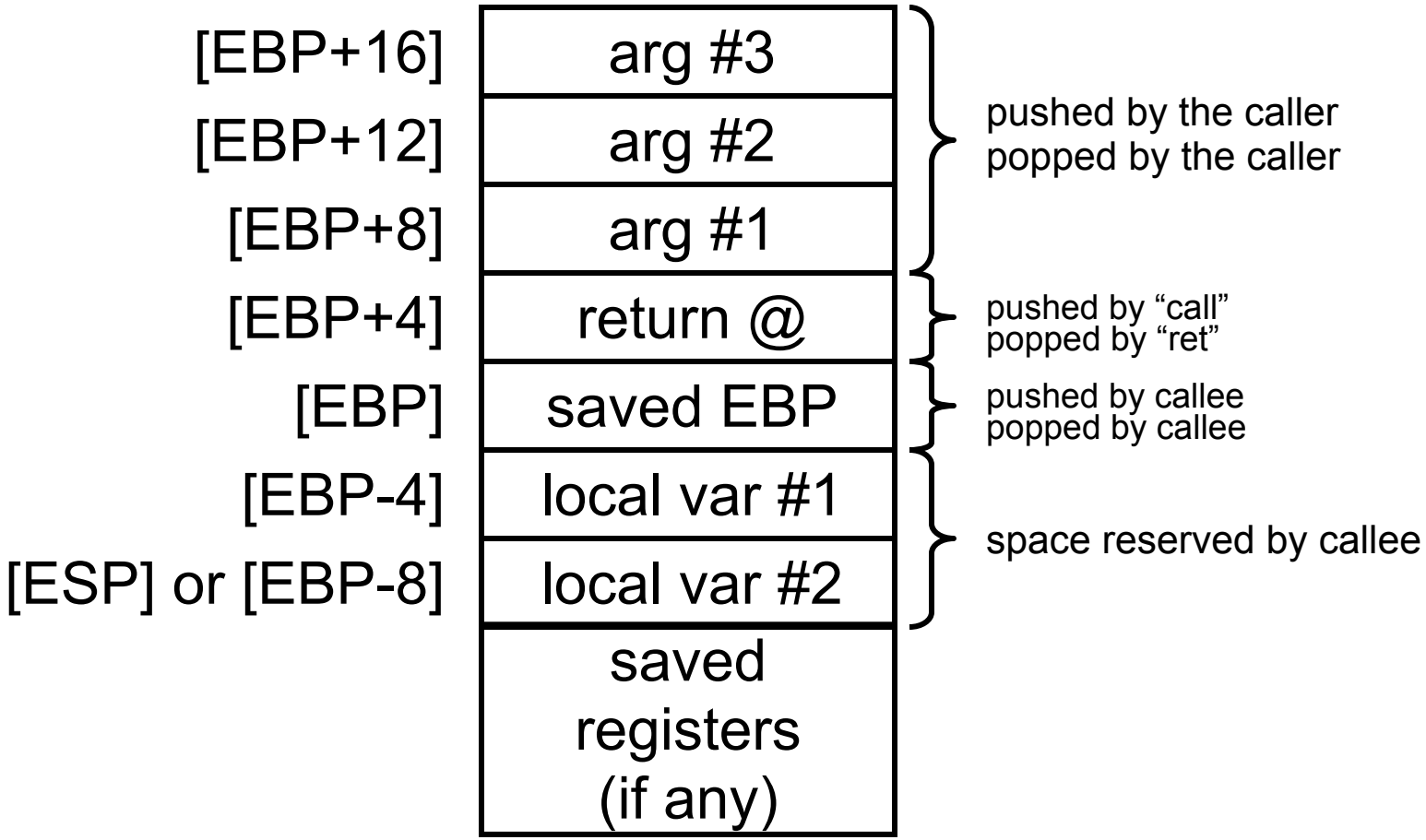
What to Study for the Final?

- Open-notes/computer, Laulima (just like the midterm)
- Non-cumulative: only post-midterm content
  - But of course you must remember pre-midterm assembly programming (e.g., when you implement a function)
- Types of questions
  - Mostly Problem-like questions
    - Write a piece of code to do something
    - What does this piece of code do?
    - What does the stack look like?
- What to study
  - All lecture notes
  - Homework assignment answers, past quizzes

“Obvious” Possible Questions

- Translate some C code into assembly
- Translate from assembly code into C
- Subprograms
  - Write a function with arguments and parameters
  - Draw a picture of the stack at some point in a program
  - Reason about how many things are on the stack
- Compiler
  - General knowledge
  - Write simple regular expressions / FSAs / CFGs

The Main Stack Structure



The Stack

- Should we go back and look at the subprogram pencil-and-paper homework assignment?
- Or make up some example on the fly?

Write a Function that …

- Write a function that takes 2 addresses as arguments and swaps their content, without destroying any register
  - watch out for indirections…
  - void f(int *x, int *y);

Write a Function that …

- Write a function that takes 2 addresses as arguments and swaps their content, without destroying any register

```
f:      push    ebp
        mov     ebp, esp
        pusha
        mov     eax, [ebp+8]      ; eax = 1st address
        mov     ebx, [eax]        ; ebx = content at 1st address
        mov     ecx, [ebp+12]     ; ecx = 2nd address
        mov     edx, [ecx]        ; edx = content at 2nd address
        mov     [eax], edx        ; store edx to 1st address
        mov     [ecx], ebx        ; store ebx to 2nd address
        popa
        pop     ebp
        ret
```

Translate this Function…

```
int  f(int x, int y) {
     int z;
     z = x + y;
     return z - 2;
}
```

Translate this Function…

```
int  f(int x, int y) {
     int z;
     z = x + y;
     return z - 2;
}
```

```
f:
push    ebp
mov     ebp, esp
sub     esp, 4

mov     eax, [ebp+8]
add     eax, [ebp+12]
mov     [ebp-4], eax
mov     eax, [ebp-4]
sub     eax, 2

add     esp, 4
pop     ebp
ret
```

Same with popa

```
int  f(int x, int y) {
     int z;
     z = x + y;
     return z - 2;
}
```

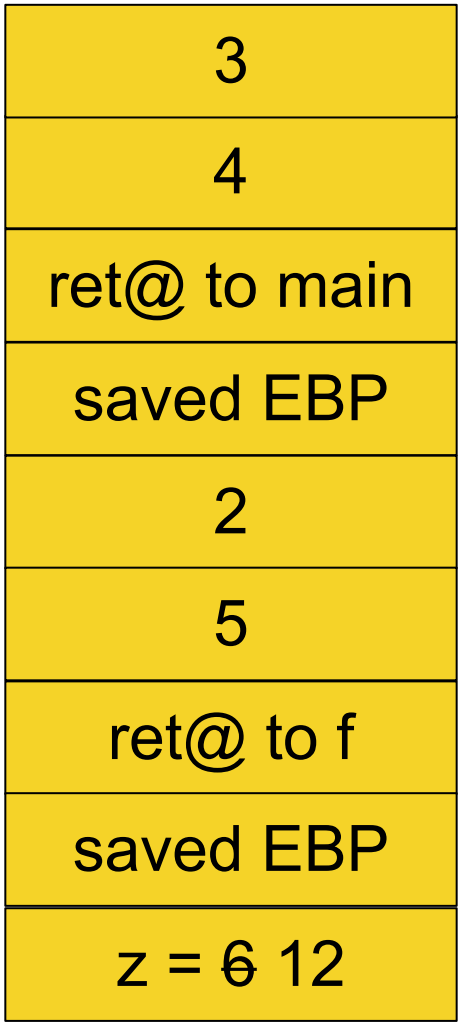
```
f:
push    ebp
mov     ebp, esp
sub     esp, 4
pusha    ; after local variables

mov     eax, [ebp+8]
add     eax, [ebp+12]
mov     [ebp-4], eax
mov     eax, [ebp-4]
sub     eax, 2
mov     [somewhere], eax

popa
mov     eax, [somewhere]
add     esp, 4
pop     ebp
ret
```

Stack picture

```
f(4,3);
....
void f(int a, int b) {
    g(a+1, b-1);
}
void g(int x, int y) {
  int z = x+1;
  z = z * y;
  // SHOW STACK
}
```

Stack picture

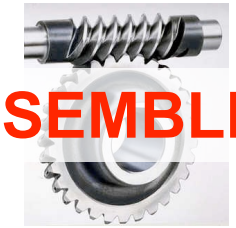
```
f(4,3);
....
void f(int a, int b) {
    g(a+1, b-1);
}
void g(int x, int y) {
  int z = x+1;
  z = z * y;
  // SHOW STACK
}
```



The Big Picture

High-level code

```
char *tmpfilename;
int num_schedulers=0;
int num_request_submitters=0;
int i,j;

if (!(f = fopen(filename,"r"))) {
  xbt_assert1(0,"Cannot open file %s",filename);
}
while(fgets(buffer,256,f)) {
  if (!strncmp(buffer,"SCHEDULER",9))
    num_schedulers++;
  if (!strncmp(buffer,"REQUESTSUBMITTER",16))
    num_request_submitters++;
}
fclose(f);
tmpfilename = strdup("/tmp/jobsimulator_
```

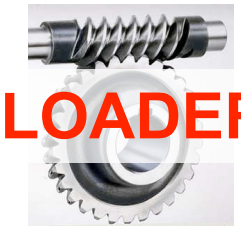


ASSEMBLER

Machine Code (object files)

```
010000101010110110
10
10  010000101010110110
10  101
11  101  010000101010110110
00  101  101010101111010101
01  111  101001010101010001
00  000  101010101010100101
    010  111100001010101001
    000  000101010111101011
         010000000010000100
         000010001000100011
```

RUNNING PROGRAM



LOADER

Machine Code (executable)

```
0100001010101101101
1010101011110101010
1010010101010100001
1010101010101001010
1111000010101010011
0001010101111010111
0100000000100001000
0000100010001000110
1010101010111011100
1010101010100100000
0000101011101011111
0010101010111111111
1111111111111010100
0101011111101101010
1101010101010101010
1111101010101010100
```

LINKER

COMPILER

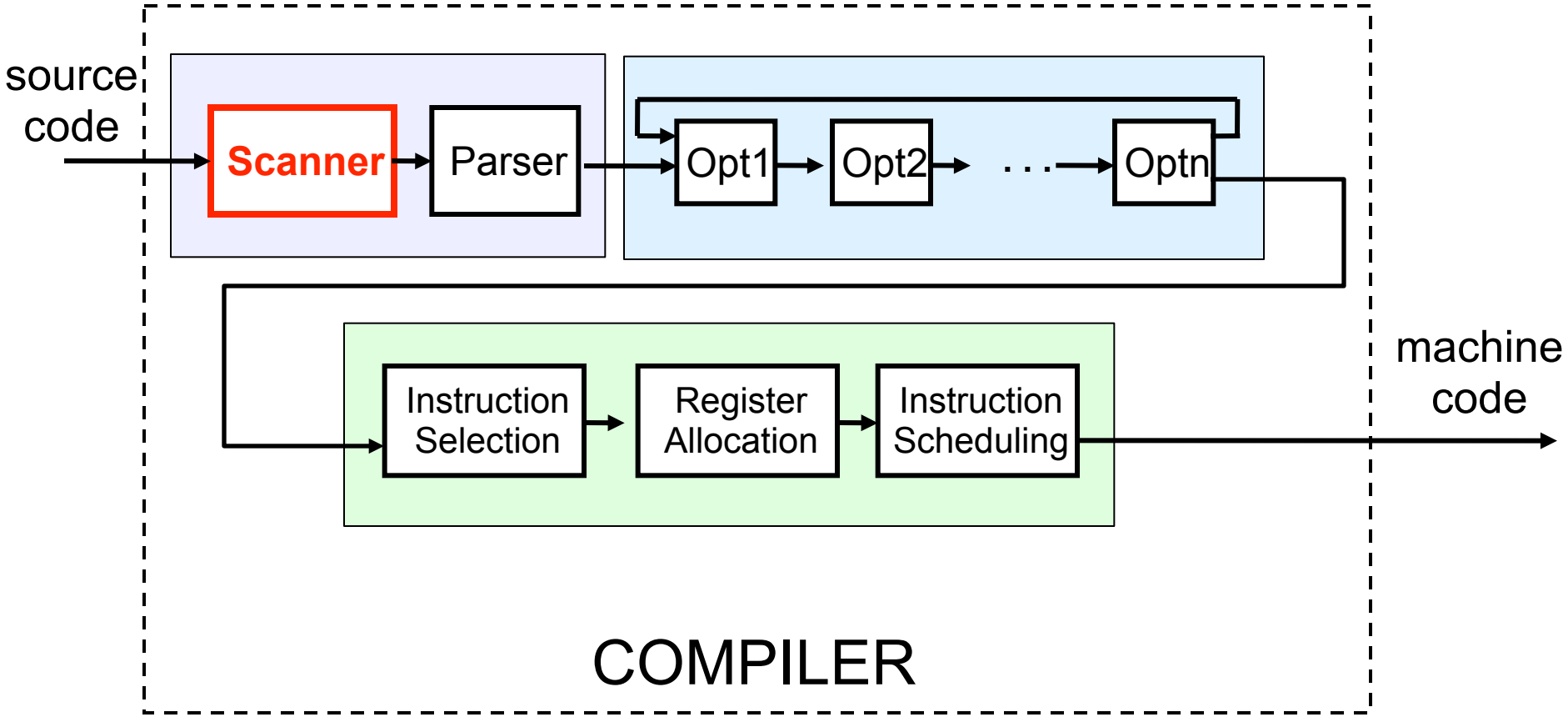
Assembly code

```
sll $t3, $t1, 2
add $t3, $s0, $t3
sll $t4, $t0, 2
add $t4, $s0, $t4
lw  $t5, 0($t3)
lw  $t6, 0($t4)
slt $t2, $t5, $t6
beq $t2, $zero, endif
add $t0, $t1, $zero
sll $t4, $t0, 2
add $t4, $s0, $t4
lw  $t5, 0($t3)
lw  $t6, 0($t4)
slt $t2, $t5, $t6
beq $t2, $zero, endif
```

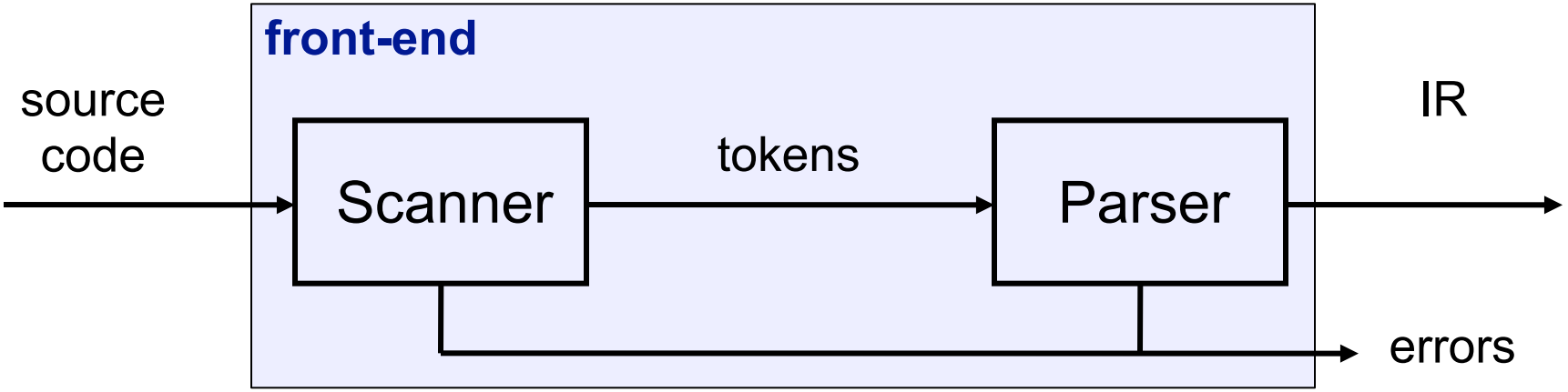
Hand-written Assembly code

```
sll $t3, $t1, 2
add $t3, $s0, $t3
sll $t4, $t0, 2
add $t4, $s0, $t4
lw  $t5, 0($t3)
lw  $t6, 0($t4)
slt $t2, $t5, $t6
beq $t2, $zero, endif
```

The Compiler's Big Picture



The front-end's big picture



- The scanner:
  - uses regular expressions (implemented as finite automata) to identify correct "words"
- The parser:
  - uses a context-free grammar to identify correct "sentences"

A few regular expressions

- What is this pattern (in english)?: $a^*b^+a^*$

- What is the pattern for all sentences over the {0,1} alphabet that start with a 0, alternate with a 1 and a 0 an arbitrary number of times, and end with any number of 0?

A few regular expressions

- What is this pattern (in english)?: $a^*b^+a^*$
  - The set of strings over alphabet {a,b} that begin with 0 or more a's, then have 1 or more b's, and then end with 0 or more a's.
  - Examples: aaaaabbbbba, ba, b, aba, abbaa

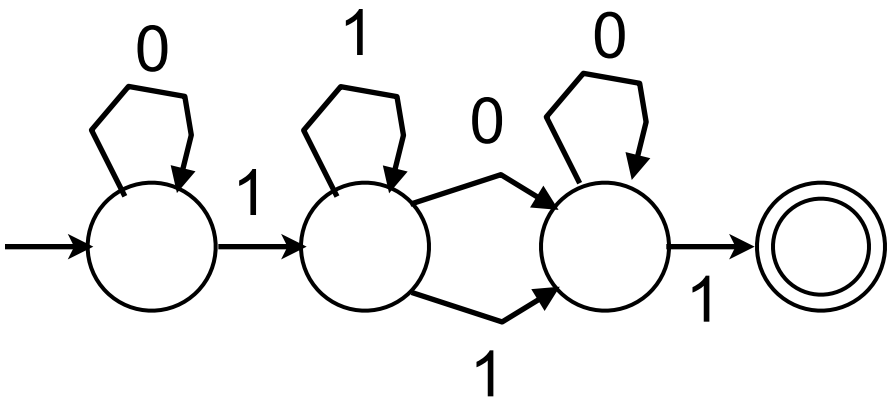
- What is the pattern for all sentences over the {0,1} alphabet that start with a 0, alternate with a 1 and a 0 an arbitrary number of times, and end with any number of 0's?
  - $0(10)^*0^*$

RE to NFA

- Draw the NFA for: $0^*1^+(0|1)0^*1$

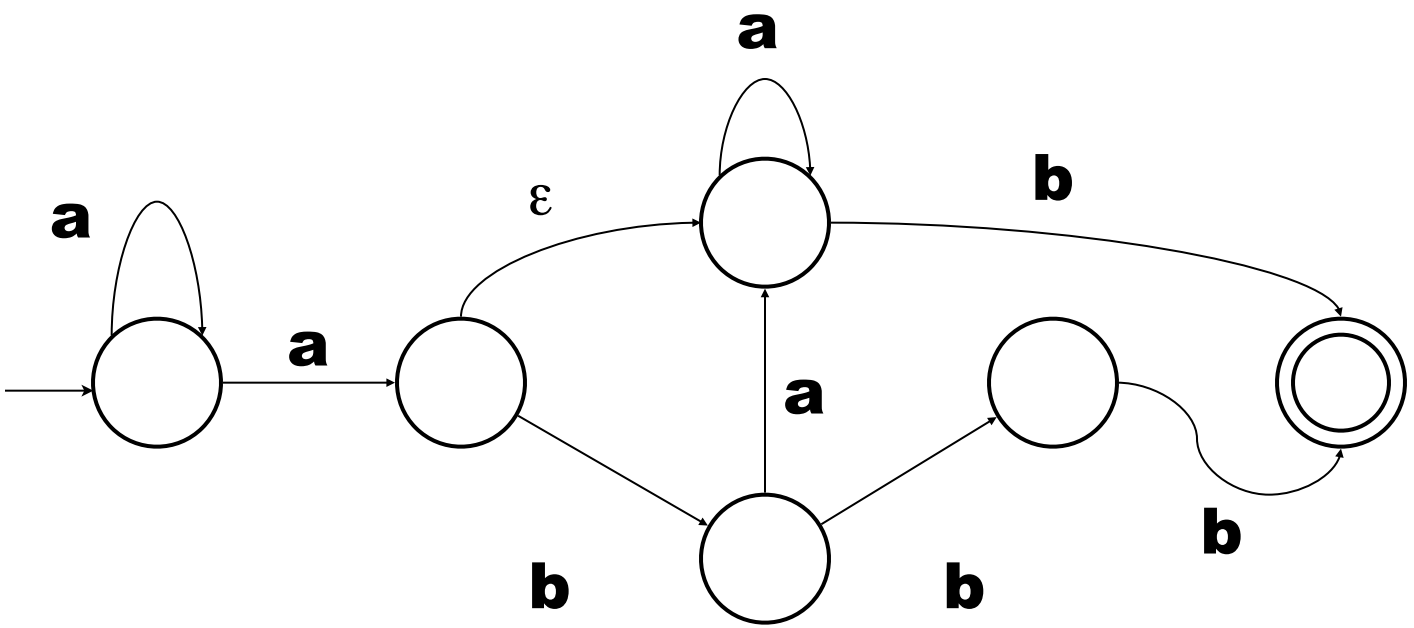
RE to NFA

- Draw the NFA for: $0^*1^+(0|1)0^*1$

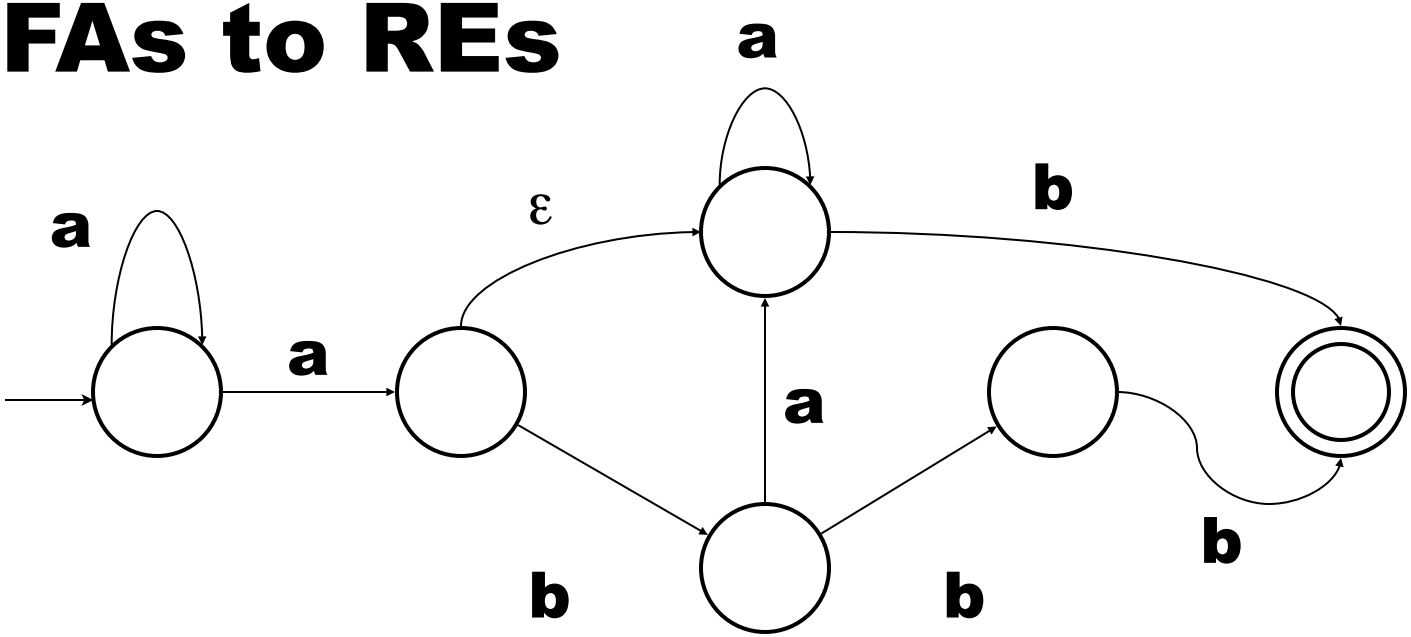


NFAs to REs

- Write the RE for this NFA



NFAs to REs



RE = $a^+a^*b \mid a^+ba^+b \mid a^+bbb$

= $a^+b \mid a^+ba^+b \mid a^+bbb$

= $a^+b(\varepsilon \mid a^+b \mid bb)$

Write a CFG

- Write the CFG over the alphabet {x,y} that describes all possible strings

Write a CFG

- Write the CFG over the alphabet {x,y} that describes all possible strings

  - $S \rightarrow \varepsilon$
  - $S \rightarrow x\ S$
  - $S \rightarrow y\ S$

Write a CFG

- Write the CFG over the alphabet {a,b} that describes all strings that are palindromes

Write a CFG

- Write the CFG over the alphabet {a,b} that describes all strings that are palindromes

  - $S \rightarrow \varepsilon \mid a \mid b$
  - $S \rightarrow a\ S\ a$
  - $S \rightarrow b\ S\ b$

Write a CFG

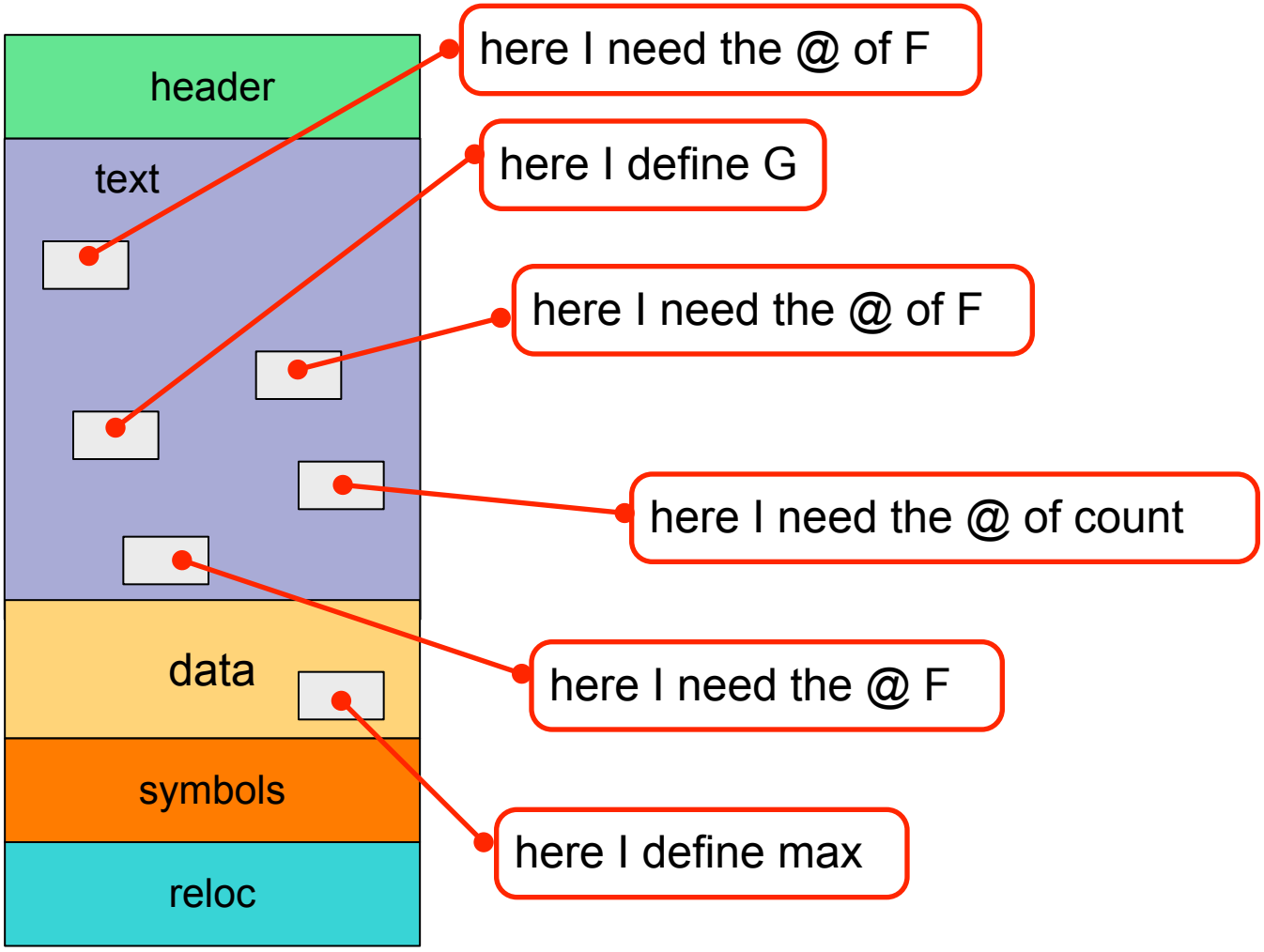
- Write the CFG over the alphabet {0,1} that describes all strings that end with a sequence of n 0's and n 1's (for any n>0)
  - example: 0101101000101010**00001111**
  - example: **000000001111111**

Write a CFG

- Write the CFG over the alphabet {0,1} that describes all strings that end with a sequence of n 0's and n 1's (for any n>0)

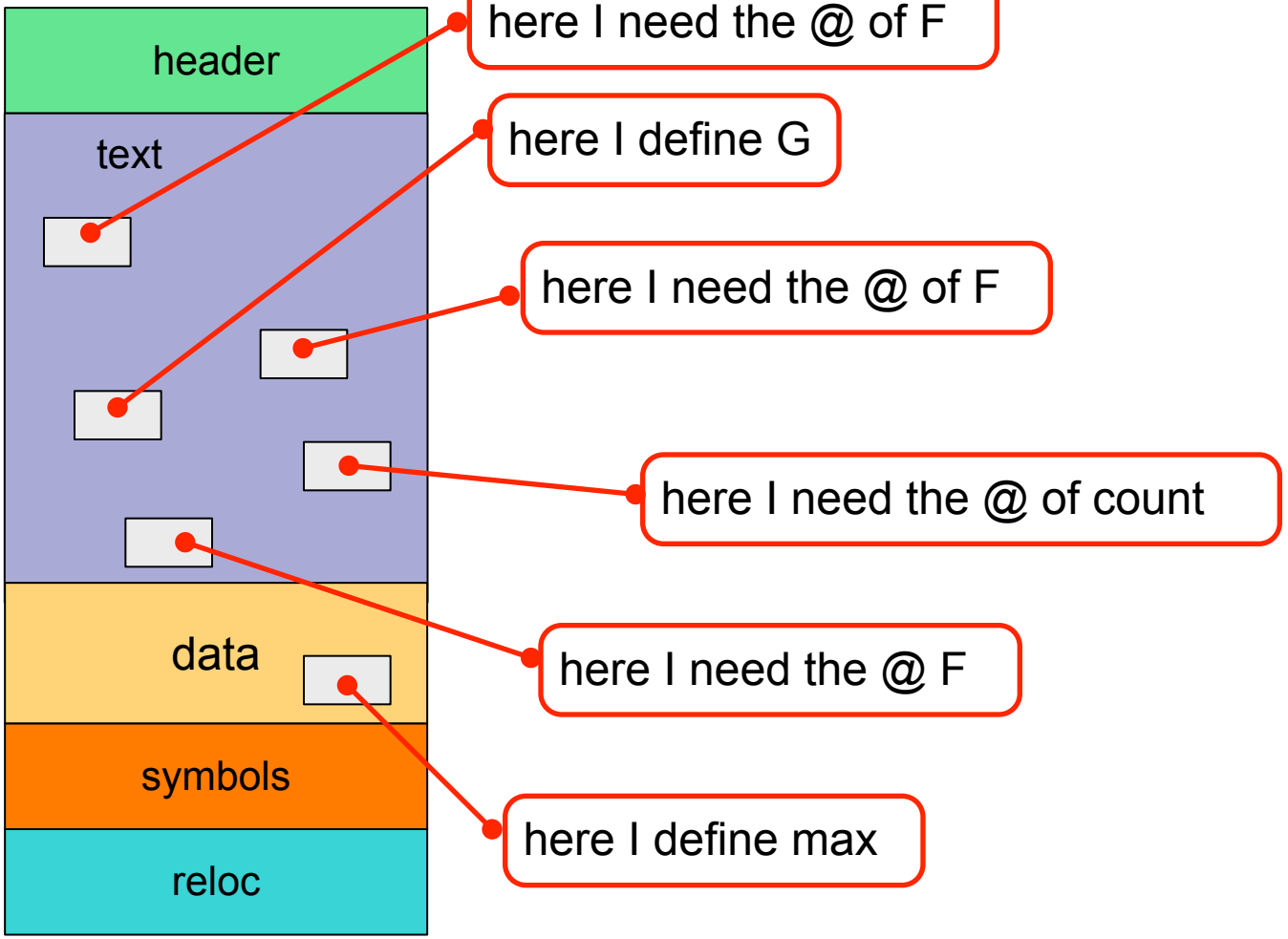
  - $S \rightarrow A\ B$
  - $A \rightarrow 0\ A \mid 1\ A \mid \varepsilon$
  - $B \rightarrow 0\ B\ 1 \mid 01$

Object files from the assembler



Object files from the assembler

how many entries in the symbol table?
how many entries in the relocation table?



The end

- Our final is Thursday December 17th at 2:15PM
  - Scheduled for 2 hours
  - Should be doable in much less