Compilation of Expressions in GCC

– lecture 4 pre-recording –

An Expression and its Translation (1)

```
int f( int a, int b, int c,
       int d, int e, int f,
       int g, int h, int i) {
    return
      (a+b)*(a-(c-b)) +
        (d+e+f*(d-e/f)+g)*h/i/a/b/c/d ;
}
```

Part 1

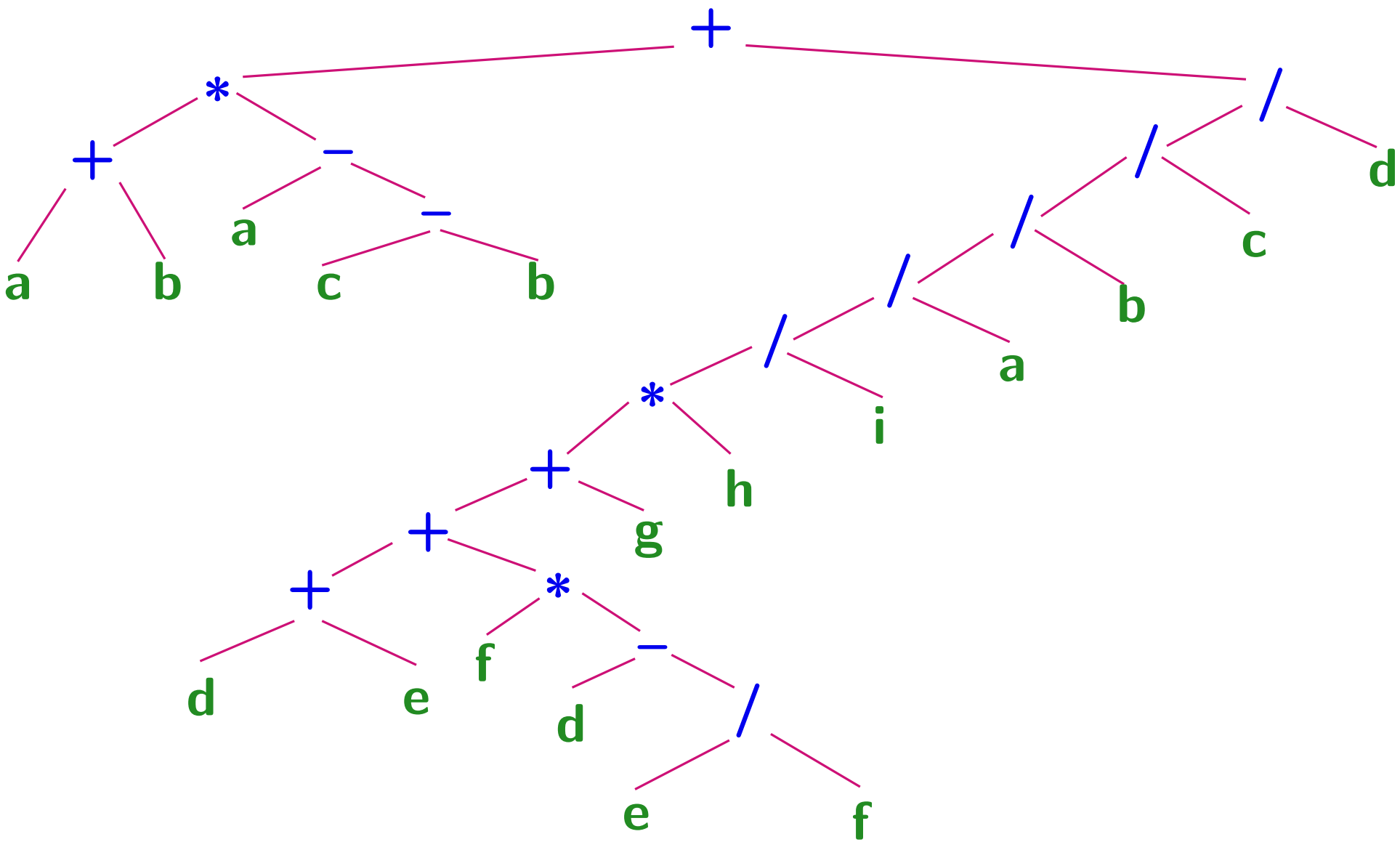
```
01. subl $12, %esp          # room to save callee saved registers
02. movl 24(%ebp), %eax     # eax = e
03. movl 20(%ebp), %ecx     # ecx = d
04. movl %edi, 8(%esp)      # save edi
05. movl 28(%ebp), %edi     # edi = f
06. movl %ebx, (%esp)       # save ebx
07. movl %eax, %edx         # edx = eax
08. sarl $31, %edx          # sign extend
09. leal (%eax,%ecx), %ebx  # ebx = eax+ecx (e+d)
10. idivl %edi              # eax /= edi    (e/f)
11. movl %ecx, %edx         # edx = ecx     (d)
12. addl 32(%ebp), %ebx     # ebx += g      (g+d+e)
13. movl %esi, 4(%esp)      # save esi
14. movl 12(%ebp), %esi     # esi = b
15. addl 8(%ebp), %esi      # esi += a      (b+a)
16. subl %eax, %edx         # edx -= eax    (d-e/f)
17. movl %edx, %eax         # eax = edx     (d-e/f)
18. imull %edi, %eax        # eax *= edi    ((d-e/f)*f)
19. movl 8(%esp), %edi      # restore edi
20. leal (%ebx,%eax), %eax  # eax = ebx+eax (e+d+g+(d-e/f)*f)
21. movl (%esp), %ebx       # restore ebx
22. imull 36(%ebp), %eax    # eax *= h      (e+d+g+(d-e/f)*f)*h
```

An Expression and Its Translation (2)

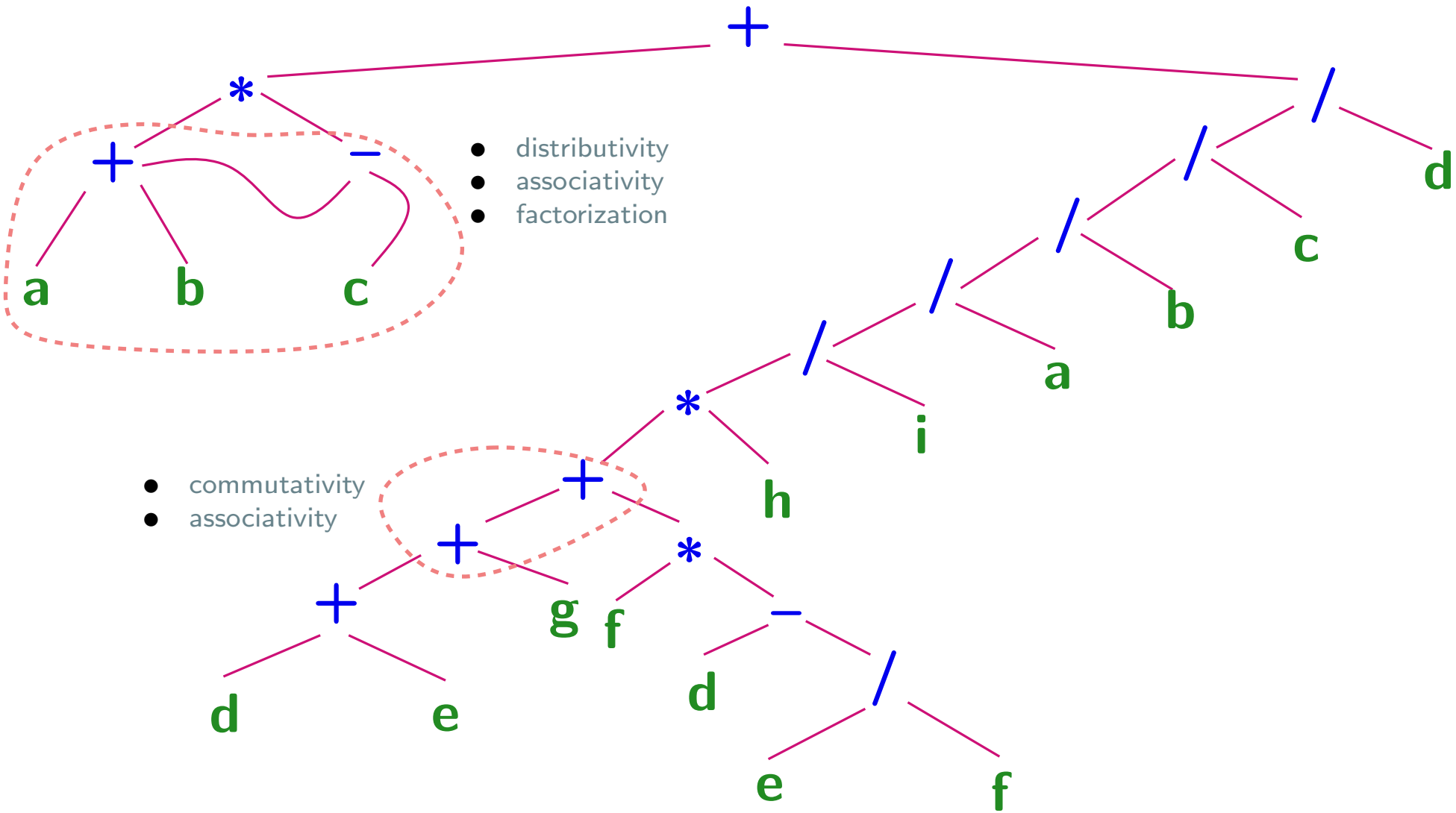
```
int f( int a, int b, int c,
       int d, int e, int f,
       int g, int h, int i) {
    return
      (a+b)*(a-(c-b)) +
        (d+e+f*(d-e/f)+g)*h/i/a/b/c/d ;
}
```

```
23. movl %eax, %edx     # edx = eax
24. sarl $31, %edx      # sign extend
25. idivl 40(%ebp)      # eax /= i      (e+d+g+(d-e/f)*f)*h/i
26. movl %eax, %edx     # edx *= eax
27. sarl $31, %edx      # sign extend
28. idivl 8(%ebp)       # eax /= a      (e+d+g+(d-e/f)*f)*h/i/a
29. movl %eax, %edx     # edx = eax
30. sarl $31, %edx      # sign extend
31. idivl 12(%ebp)      # eax /= b      (e+d+g+(d-e/f)*f)*h/i/a/b
32. movl %eax, %edx     # edx = eax
33. sarl $31, %edx      # sign extend
34. idivl 16(%ebp)      # eax /= c      (e+d+g+(d-e/f)*f)*h/i/a/b/c
35. movl %eax, %edx     # edx = eax
36. sarl $31, %edx      # sign extend
37. idivl %ecx          # eax /= ecx    (e+d+g+(d-e/f)*f)*h/i/a/b/c/d
38. movl %esi, %edx     # edx = esi
39. subl 16(%ebp), %edx # edx -= c      (a+b-c)
40. imull %esi, %edx    # edx *= esi    (a+b)*(a+b-c)
41. movl 4(%esp), %esi  # restore esi
42. addl %edx, %eax     # eax =  (a+b)*(a+b-c)+(e+d+g+(d-e/f)*f)*h/i/a/b/c/d
```

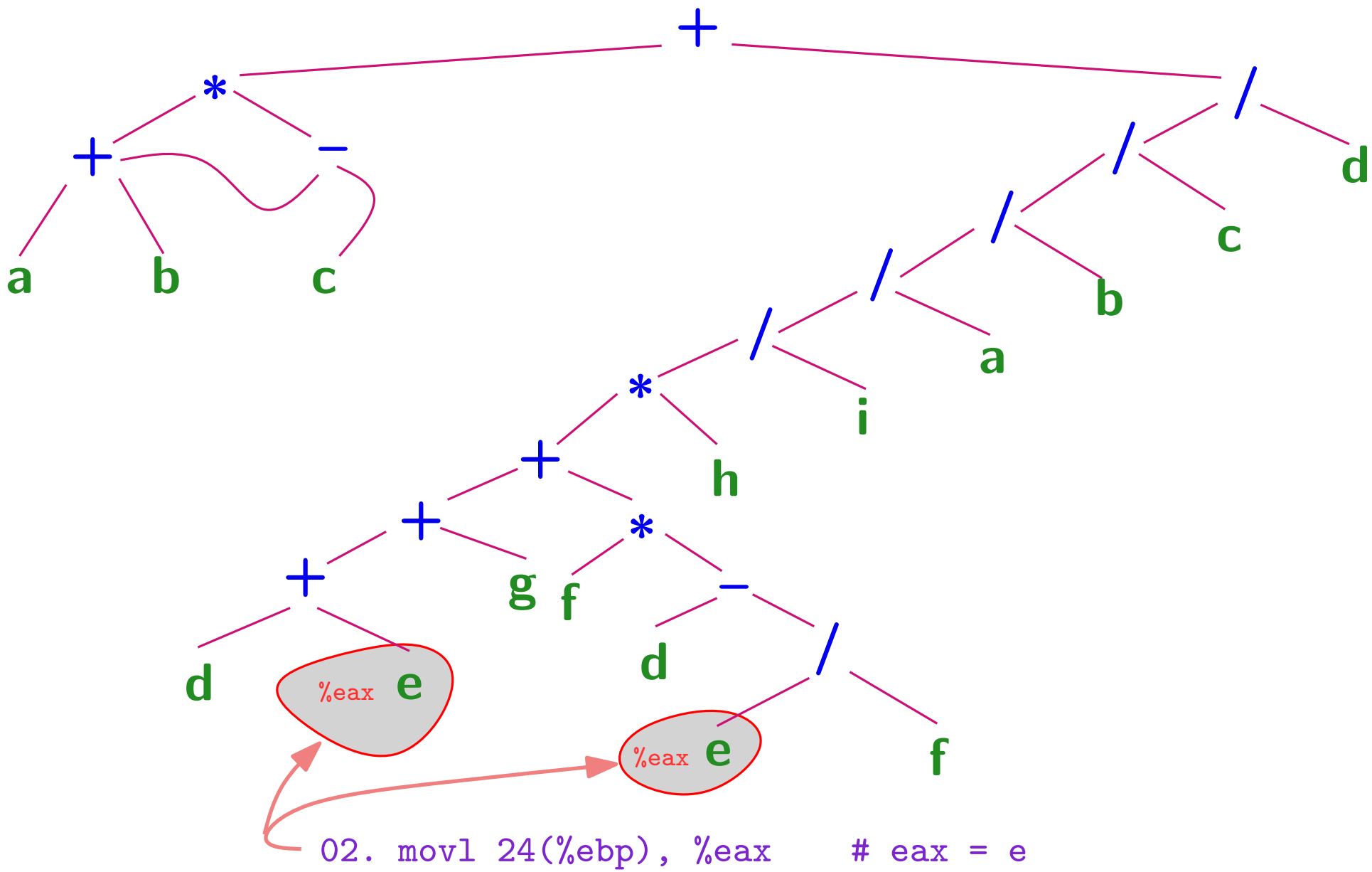
AST and its Compilation Scheme



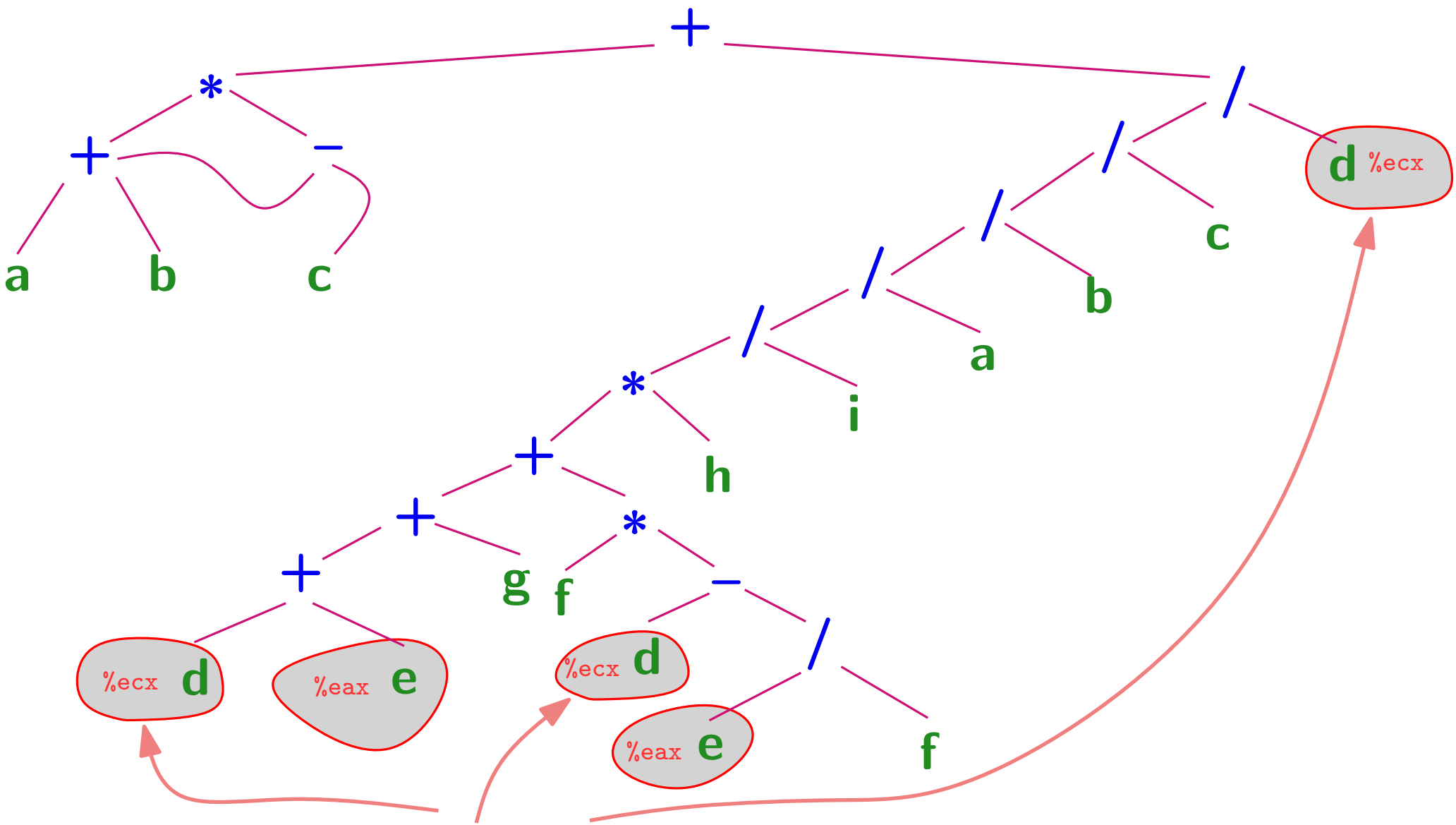
AST and its Compilation Scheme



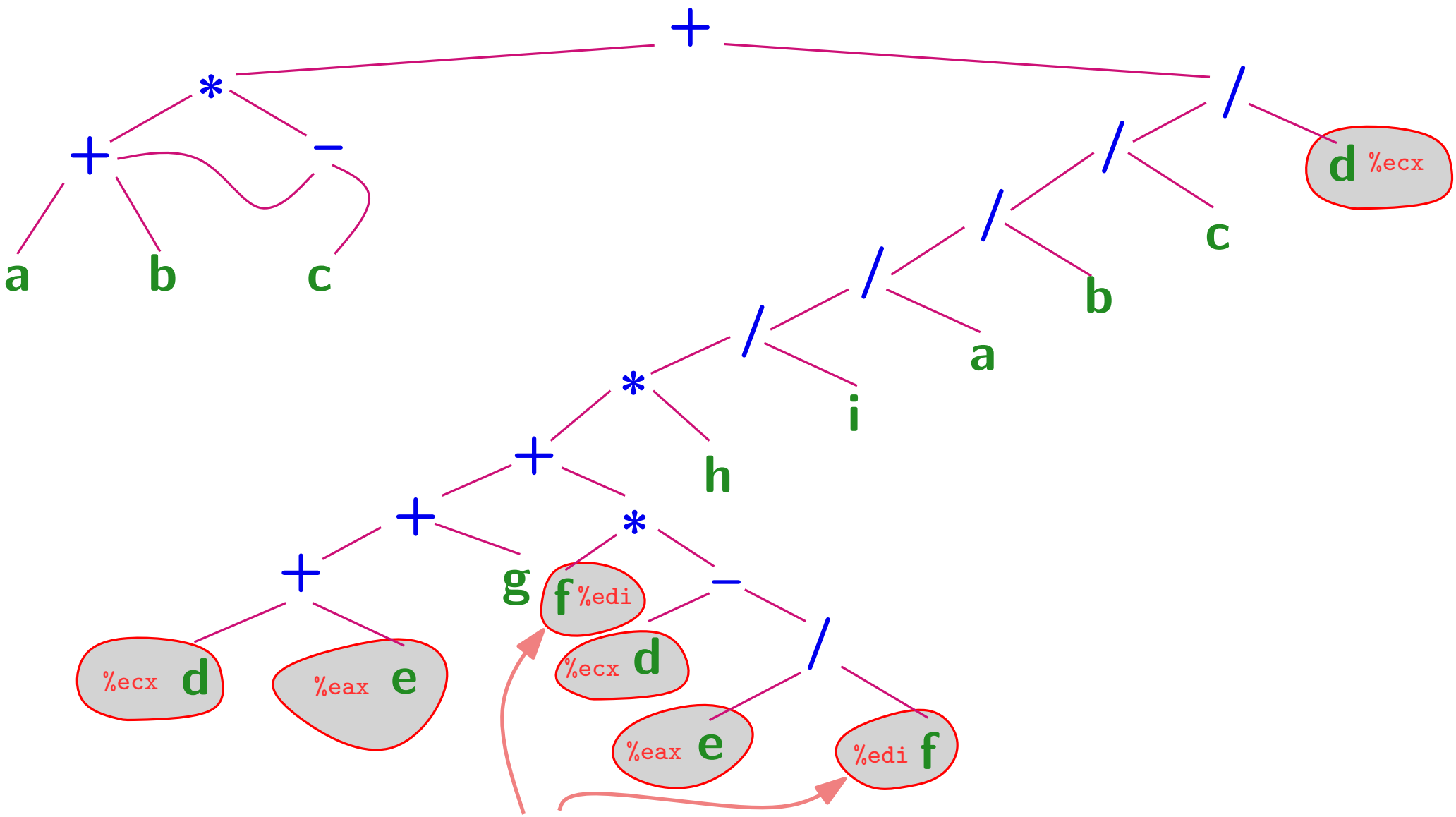
AST and its Compilation Scheme



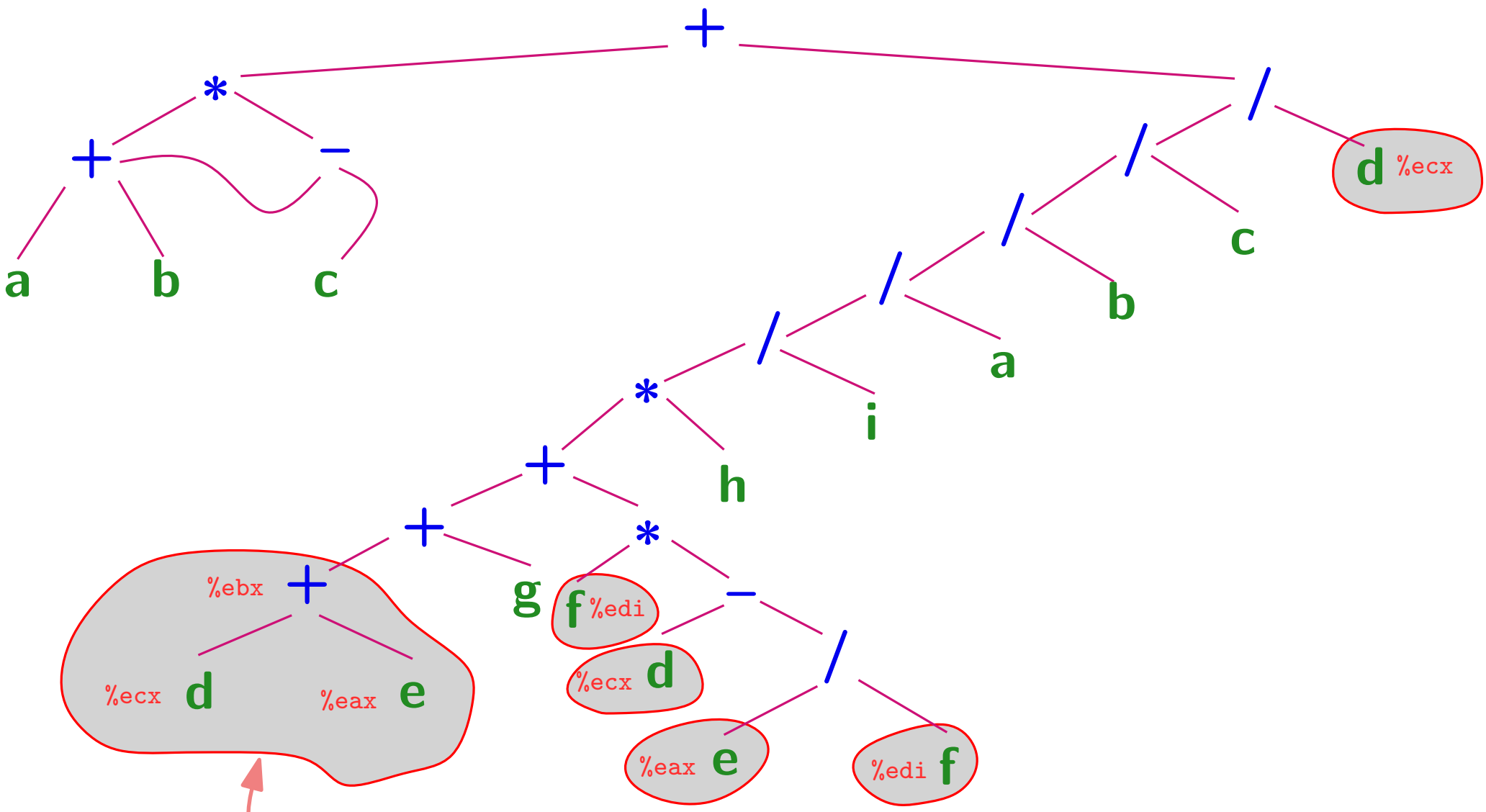
AST and its Compilation Scheme



AST and its Compilation Scheme

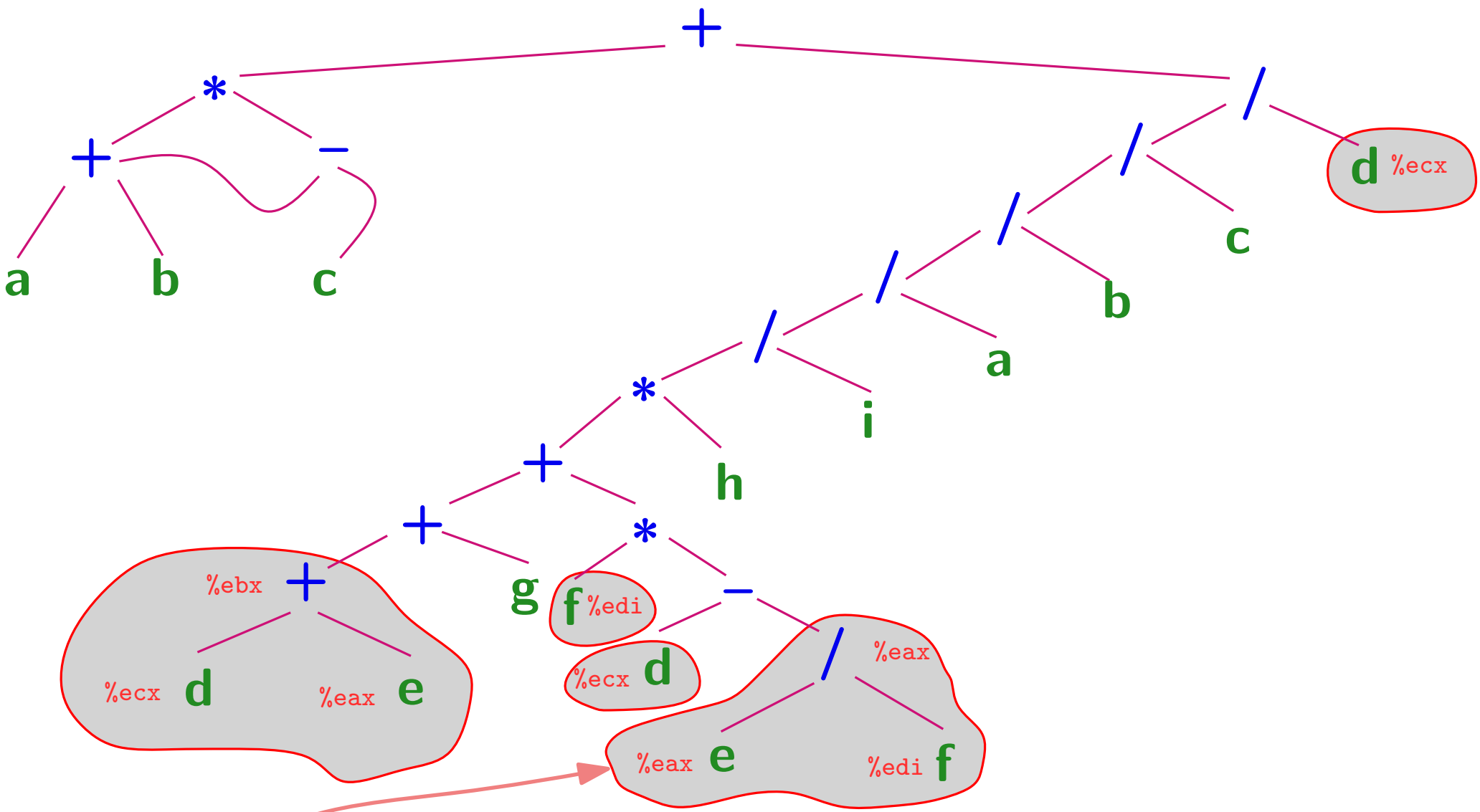


AST and its Compilation Scheme



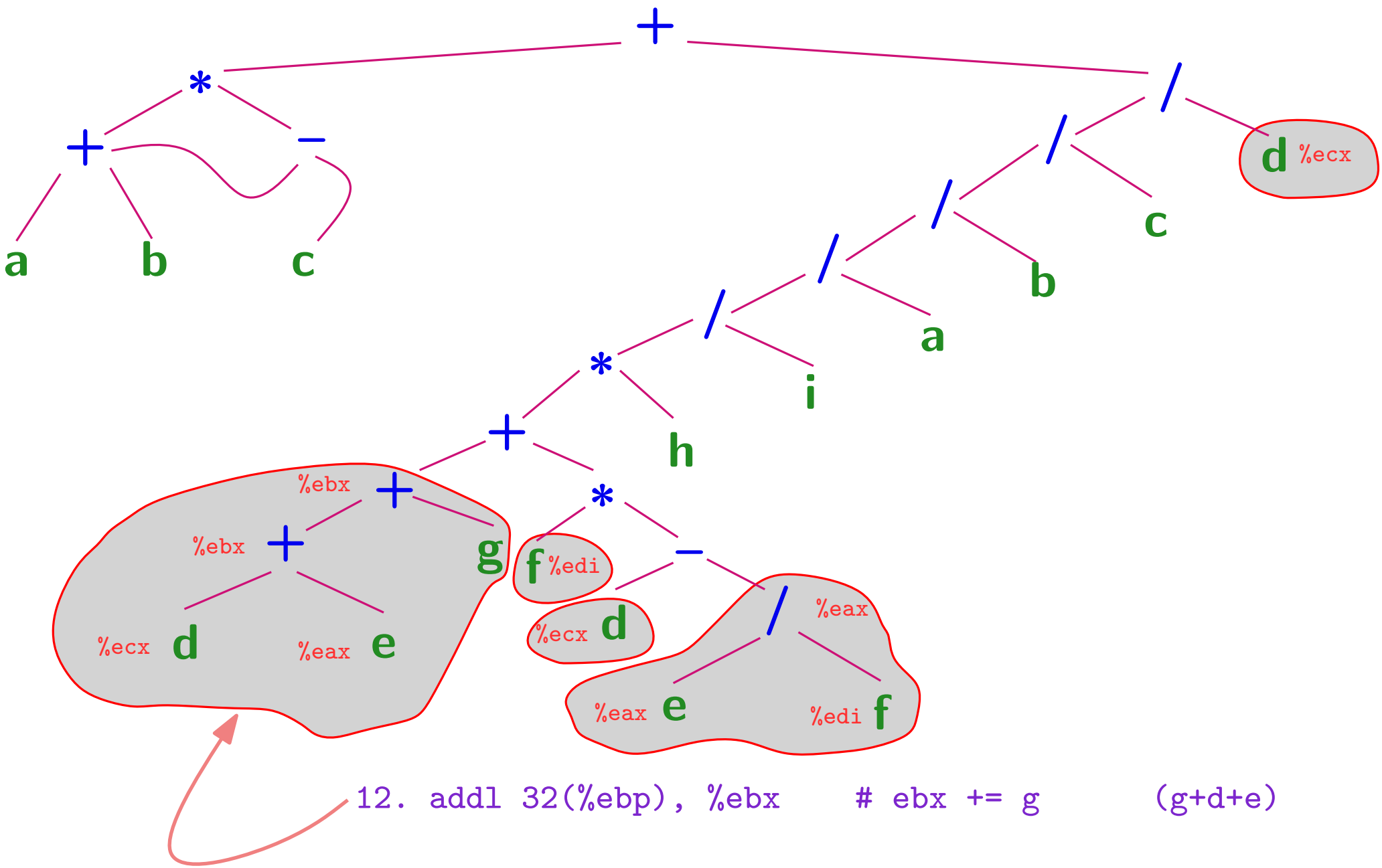

```
09. leal (%eax,%ecx), %ebx # ebx = eax+ecx (e+d)
```

AST and its Compilation Scheme



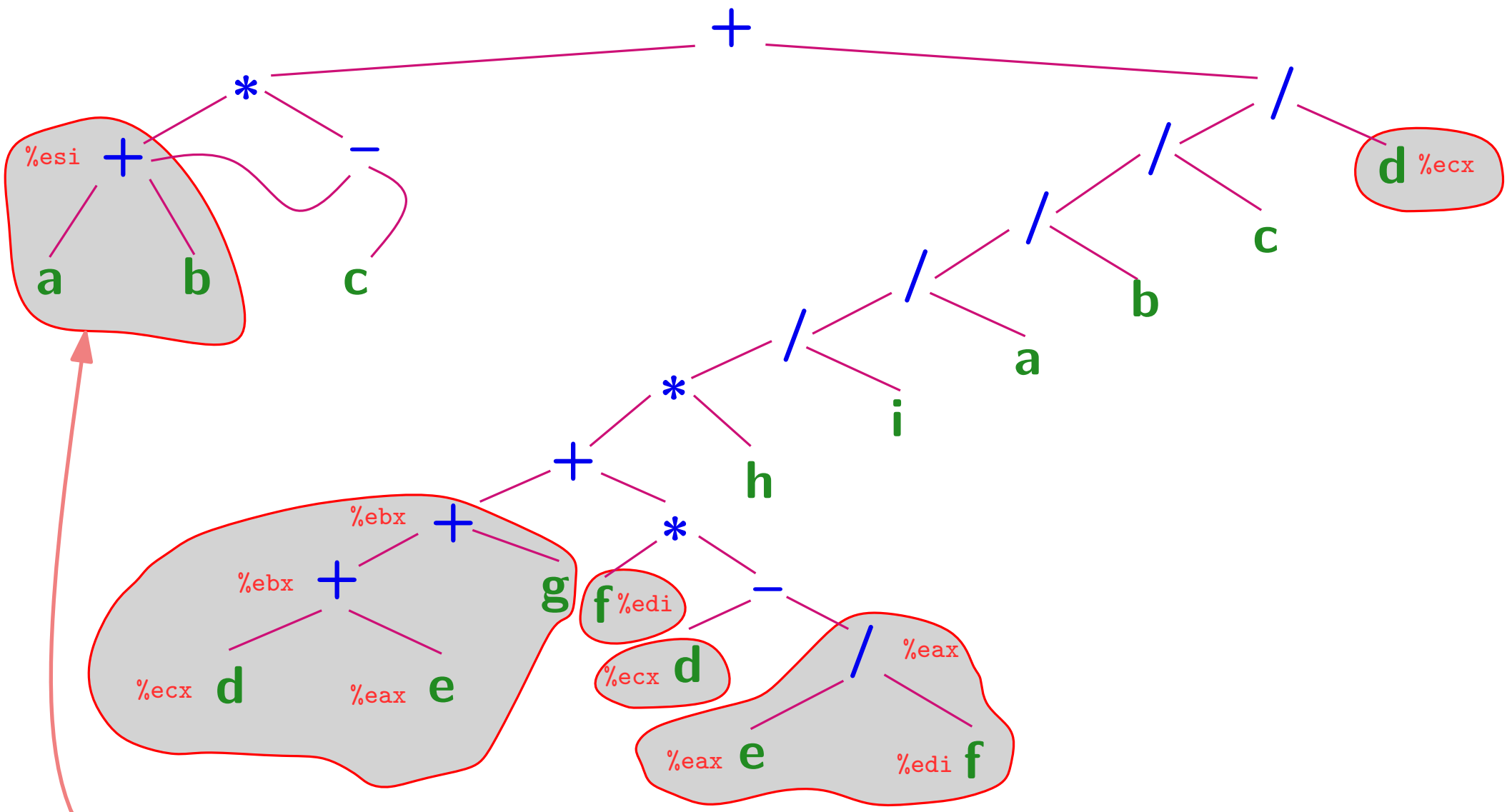

```
07. movl %eax, %edx          # edx = eax
08. sarl $31, %edx           # sign extend
10. idivl %edi               # eax /= edi    (e/f)
```

AST and its Compilation Scheme



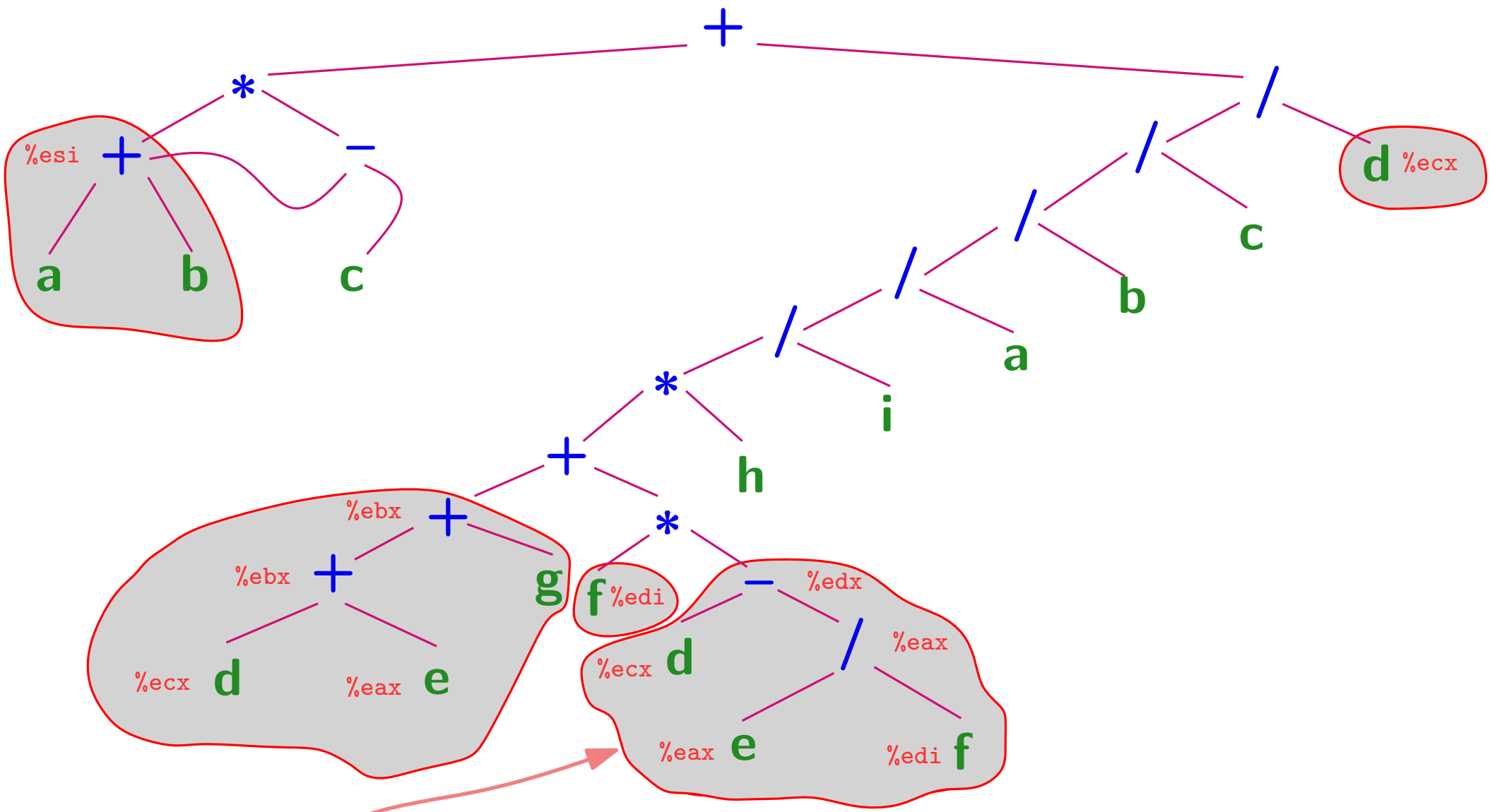

```
12. addl 32(%ebp), %ebx    # ebx += g       (g+d+e)
```

AST and its Compilation Scheme



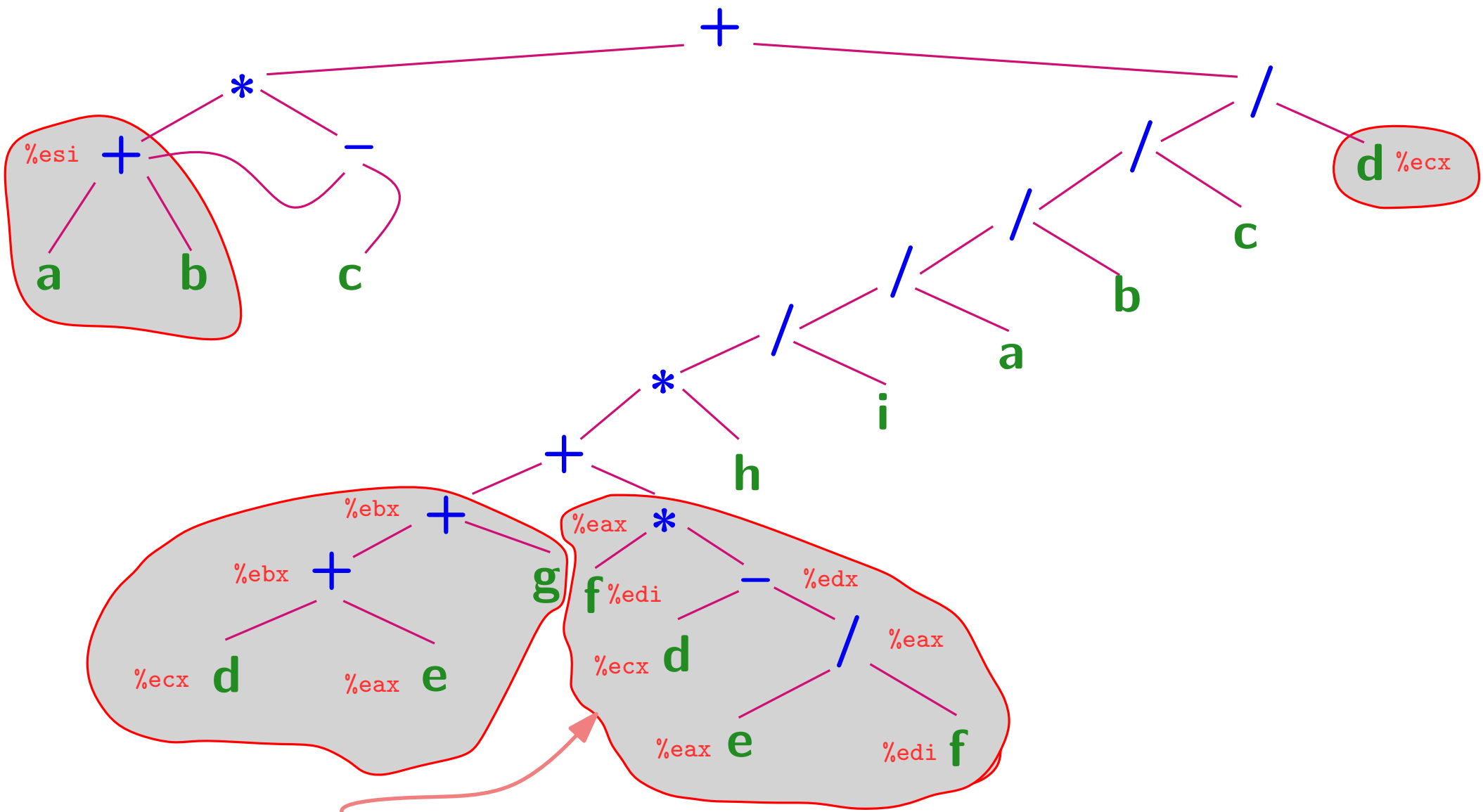

```
14. movl 12(%ebp), %esi     # esi = b
15. addl 8(%ebp), %esi      # esi += a        (b+a)
```

AST and its Compilation Scheme

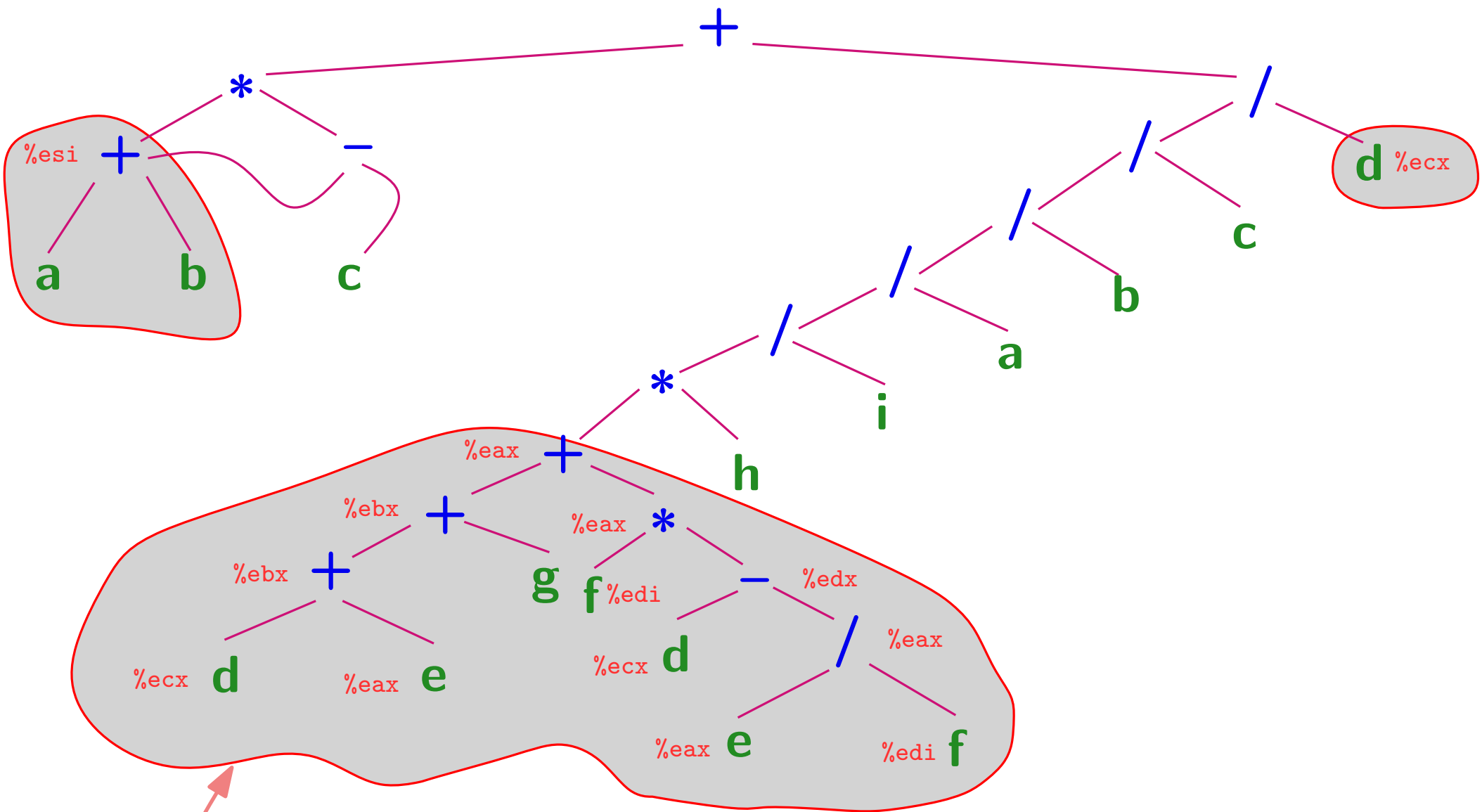



```
11. movl %ecx, %edx          # edx = ecx      (d)
16. subl %eax, %edx          # edx -= eax     (d-e/f)
17. movl %edx, %eax          # eax = edx      (d-e/f)
```

AST and its Compilation Scheme

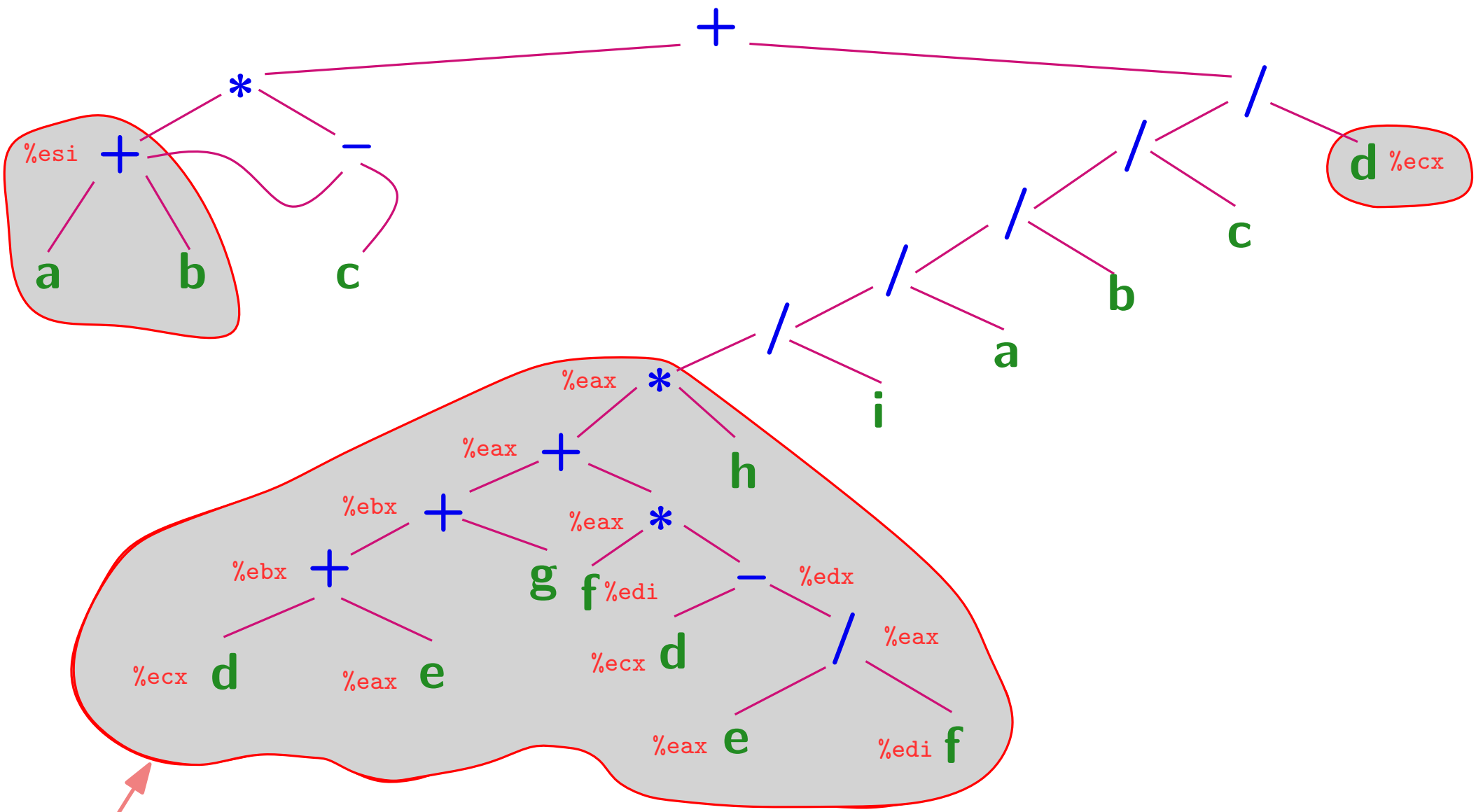


AST and its Compilation Scheme



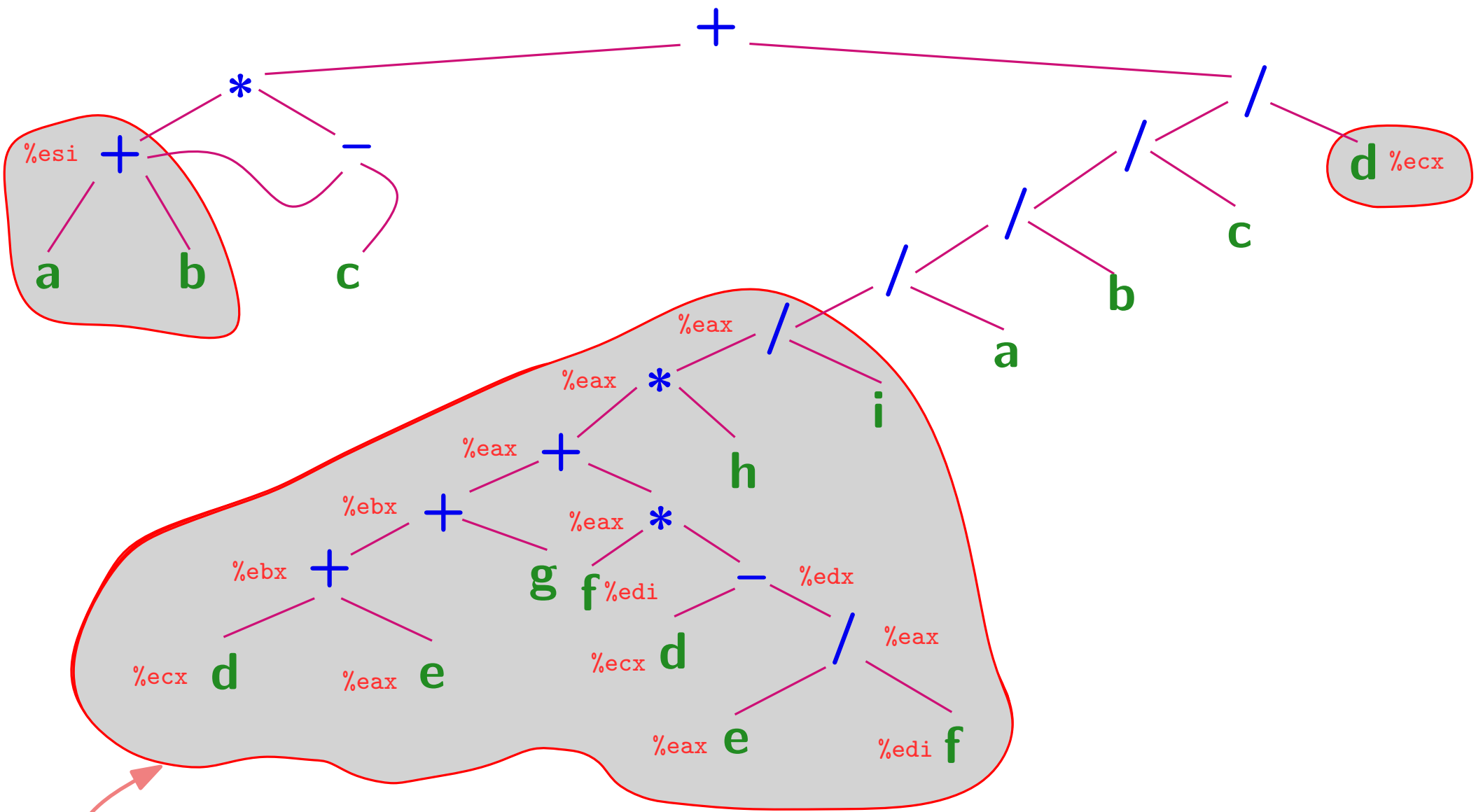

```
20. leal (%ebx,%eax), %eax # eax = ebx+eax (e+d+g+(d-e/f)*f)
```

AST and its Compilation Scheme




```
22. imull 36(%ebp), %eax   # eax *= h        (e+d+g+(d-e/f)*f)*h
```

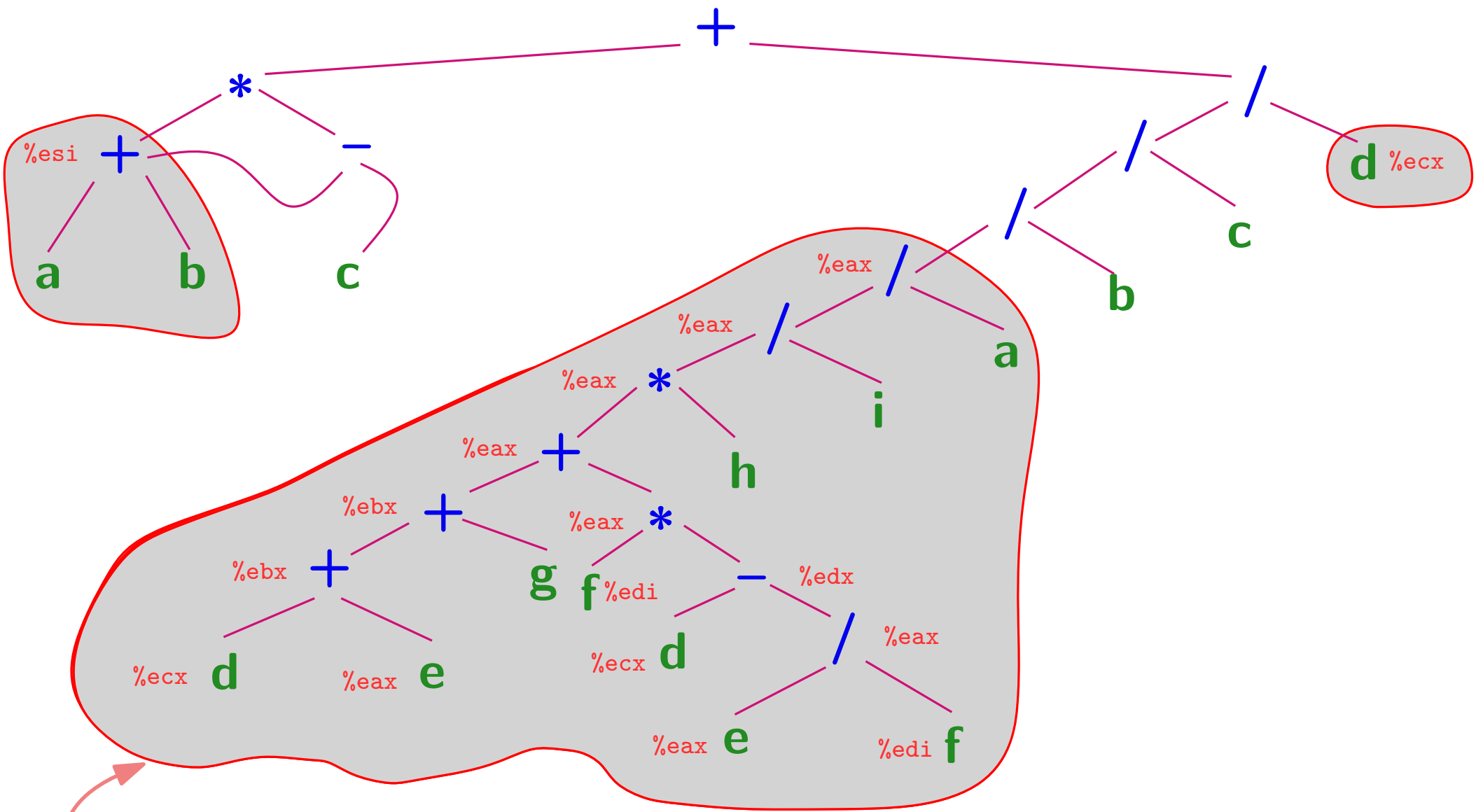
AST and its Compilation Scheme




```
23. movl %eax, %edx      # edx = eax
24. sarl $31, %edx       # sign extend
25. idivl 40(%ebp)       # eax /= i
```

(e+d+g+(d-e/f)*f)*h/i

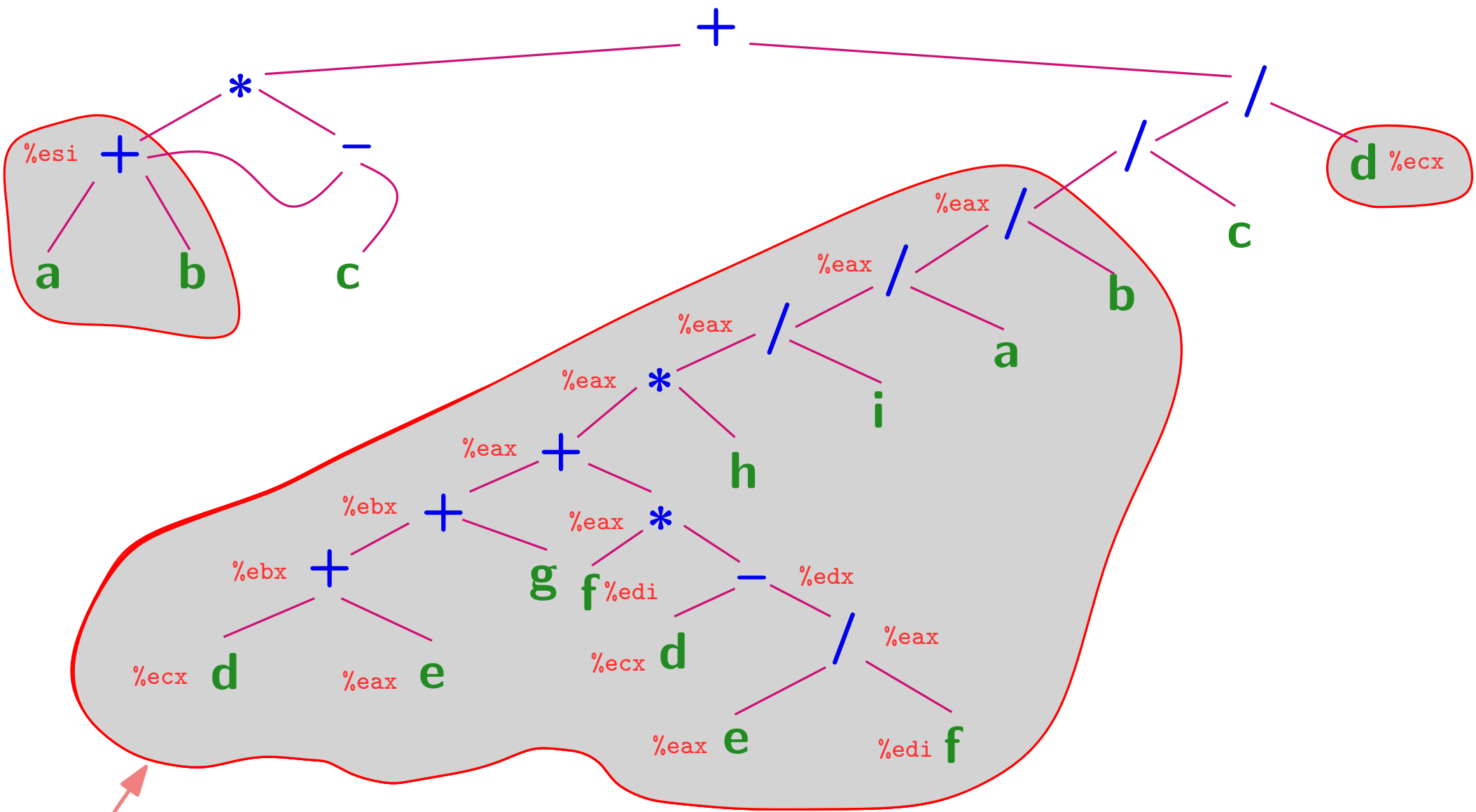
AST and its Compilation Scheme




```
26. movl %eax, %edx      # edx *= eax
27. sarl $31, %edx       # sign extend
28. idivl 8(%ebp)        # eax /= a
```

(e+d+g+(d-e/f)*f)*h/i/a

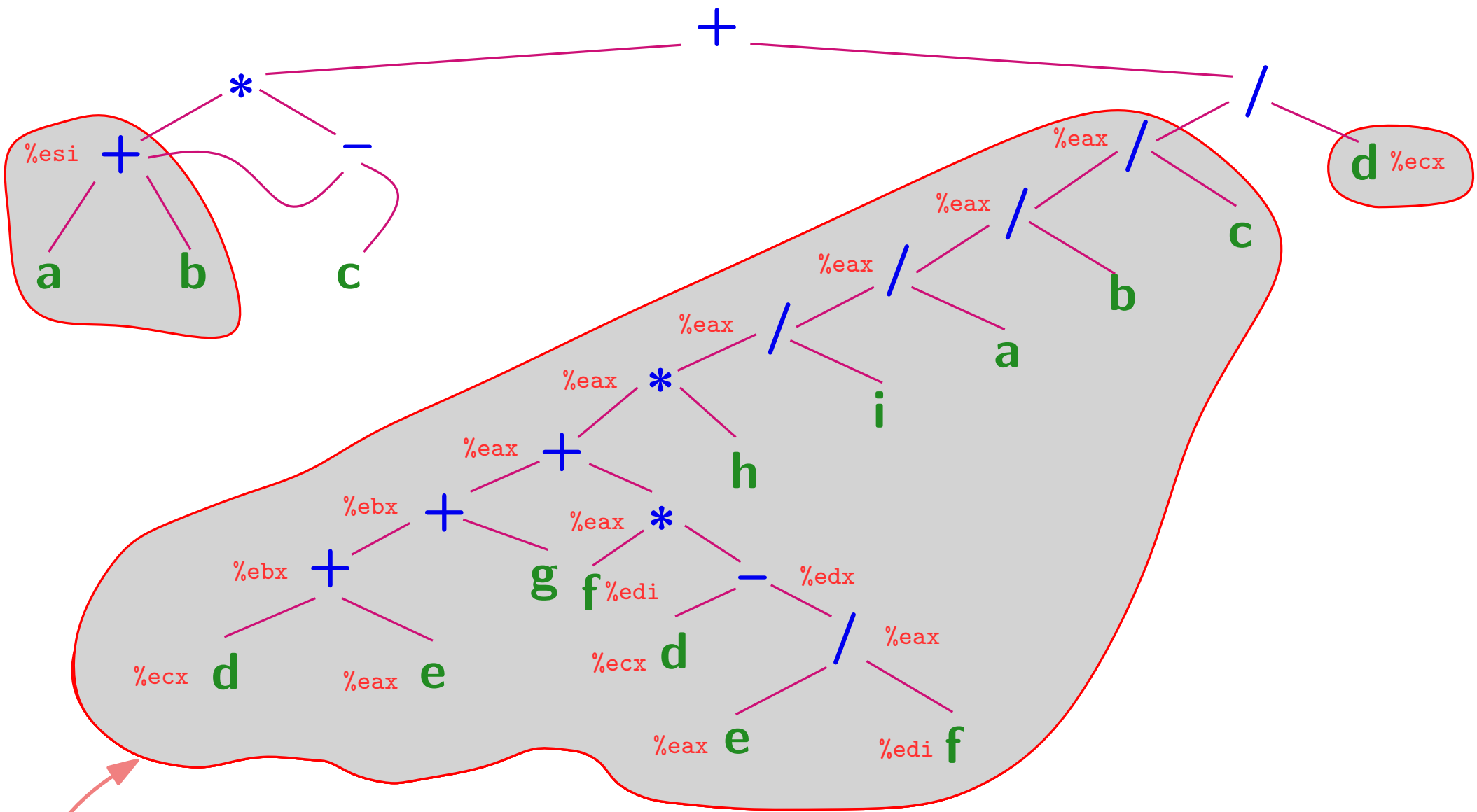
AST and its Compilation Scheme




```
29. movl %eax, %edx      # edx = eax
30. sarl $31, %edx       # sign extend
31. idivl 12(%ebp)       # eax /= b
```

(e+d+g+(d-e/f)*f)*h/i/a/b

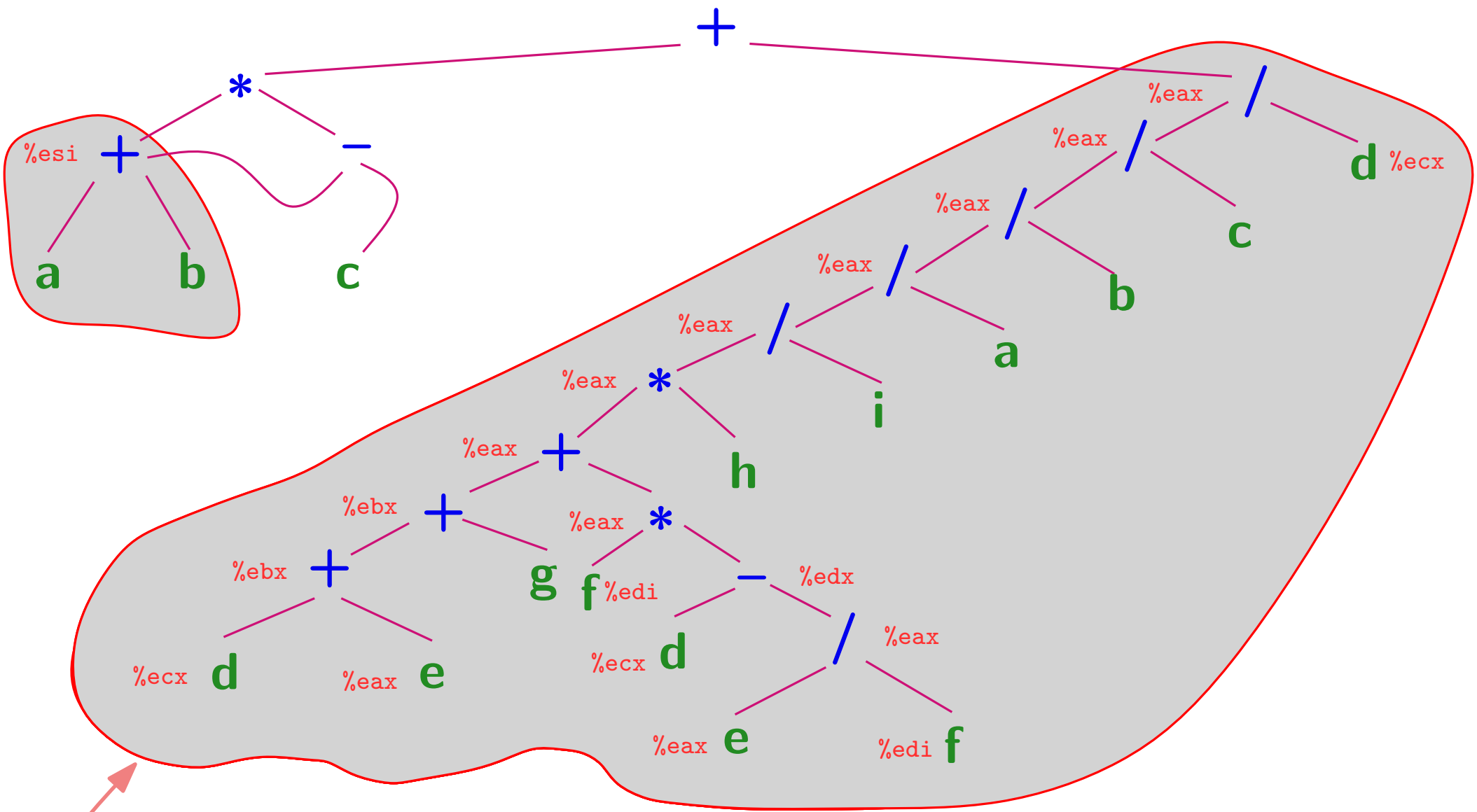
AST and its Compilation Scheme




```
32. movl %eax, %edx      # edx = eax
33. sarl $31, %edx       # sign extend
34. idivl 16(%ebp)       # eax /= c
```

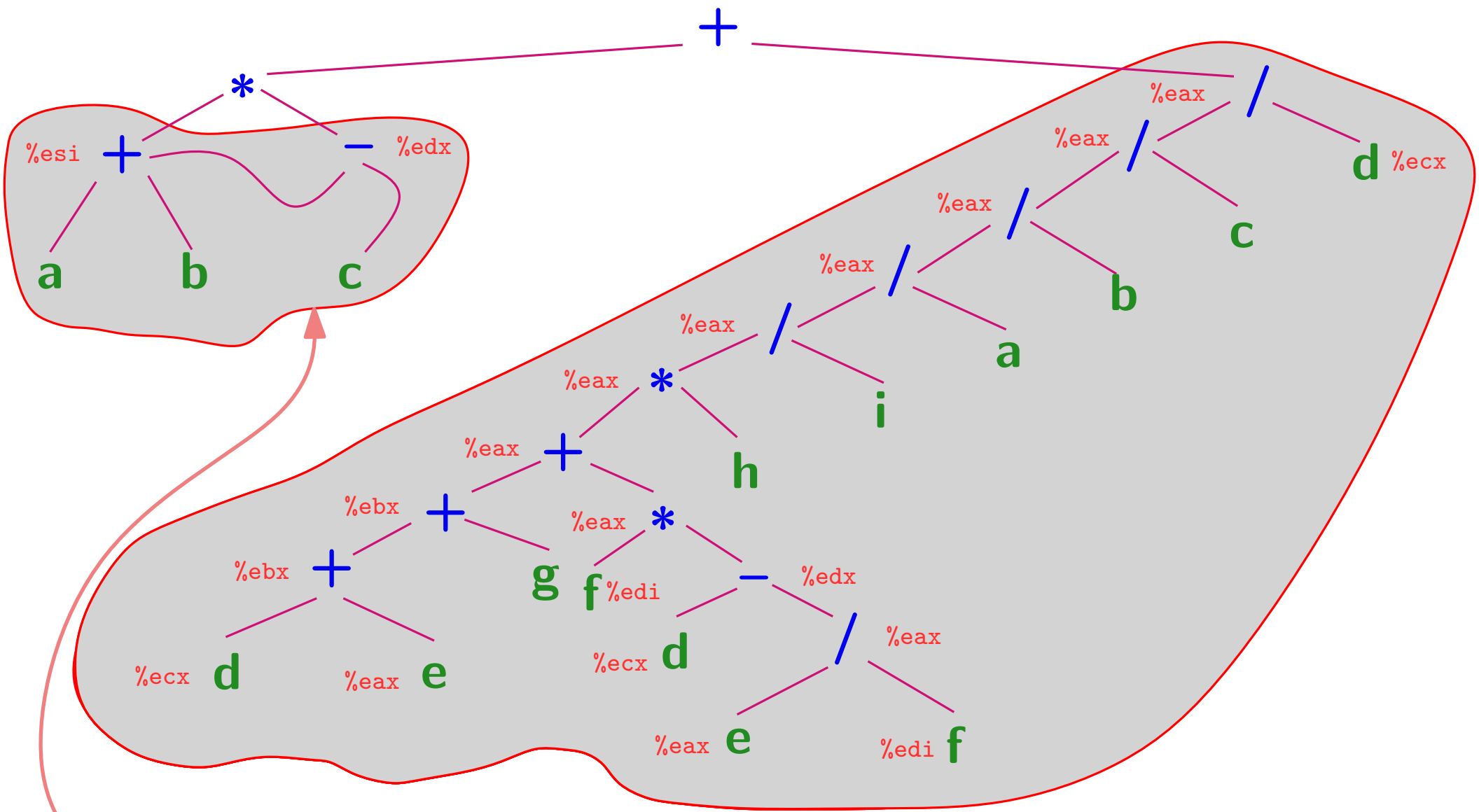
(e+d+g+(d-e/f)*f)*h/i/a/b/c

AST and its Compilation Scheme



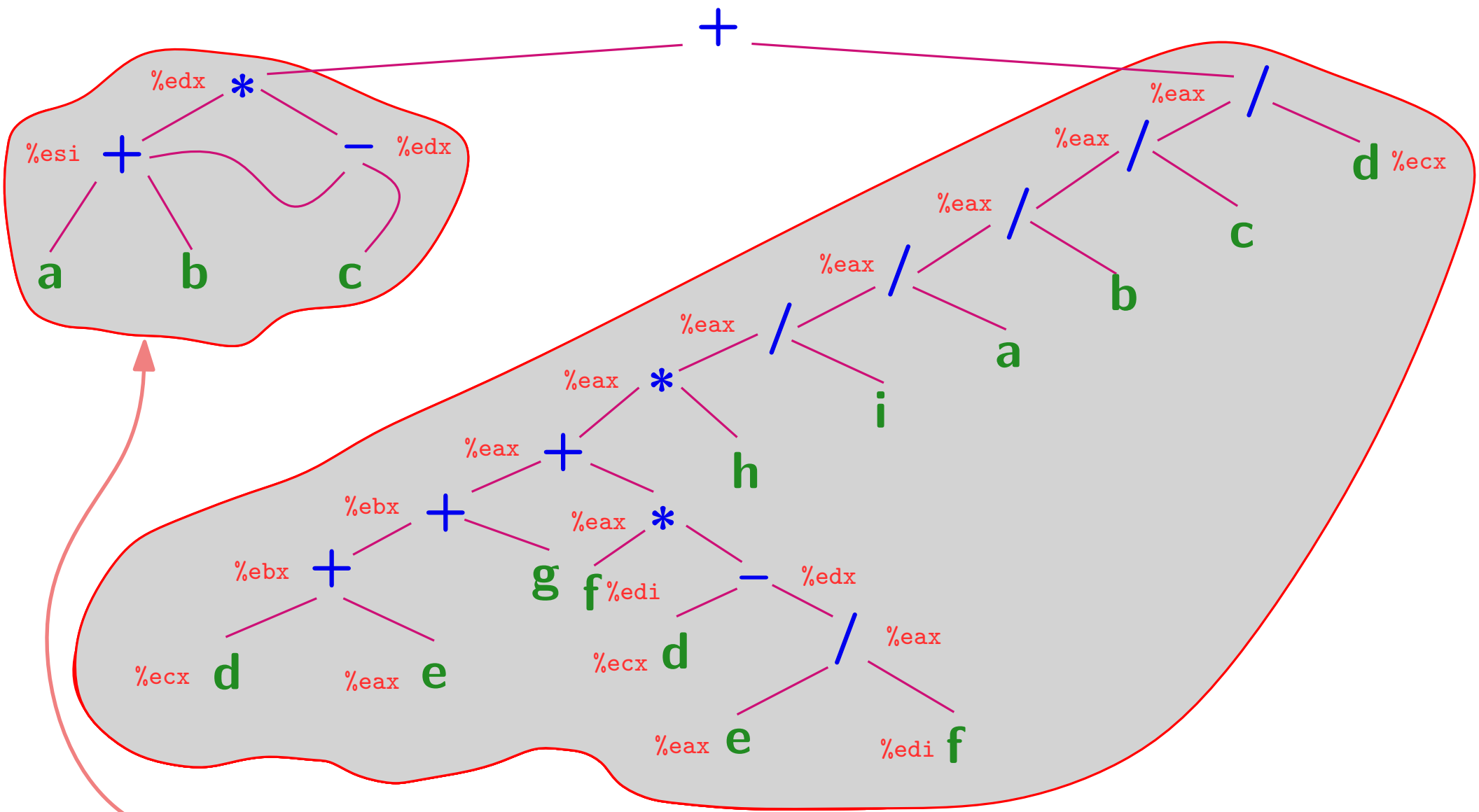

```
35. movl %eax, %edx      # edx = eax
36. sarl $31, %edx       # sign extend
37. idivl %ecx           # eax /= ecx     (e+d+g+(d-e/f)*f)*h/i/a/b/c/d
```

AST and its Compilation Scheme



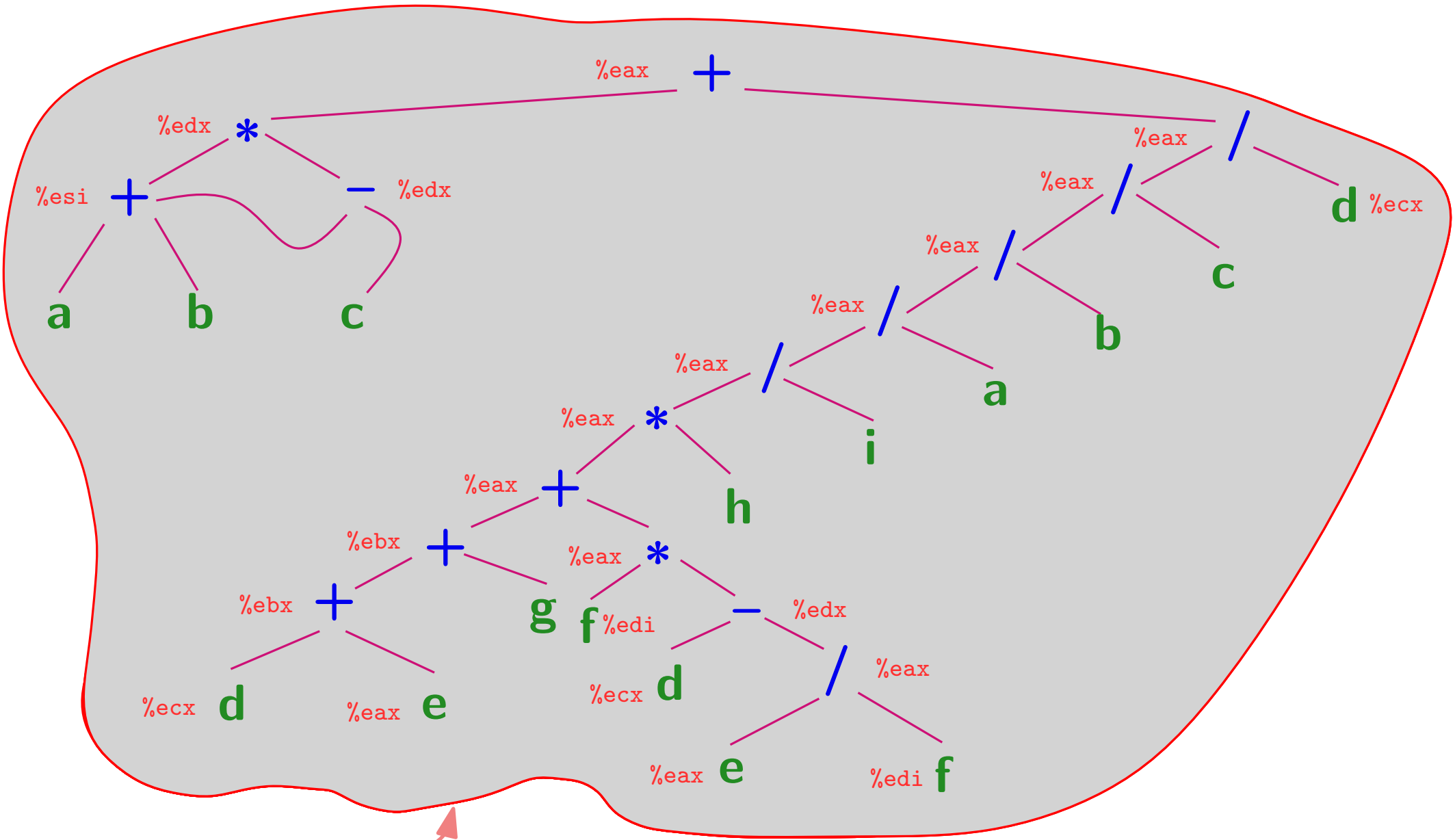

```
38. movl %esi, %edx      # edx = esi
39. subl 16(%ebp), %edx # edx -= c      (a+b-c)
```

AST and its Compilation Scheme




```
40. imull %esi, %edx    # edx *= esi    (a+b)*(a+b-c)
```

AST and its Compilation Scheme




```
42. addl %edx, %eax      # eax =  (a+b)*(a+b-c)+(e+d+g+(d-e/f)*f)*h/i/a/b/c/d
```

Compilation Scheme

- Partial results are held in registers, rather than the stack.
- Optimizations are applied to reduce the number of operations.
- Instructions are reordered to take advantage of instruction-level-parallelism.