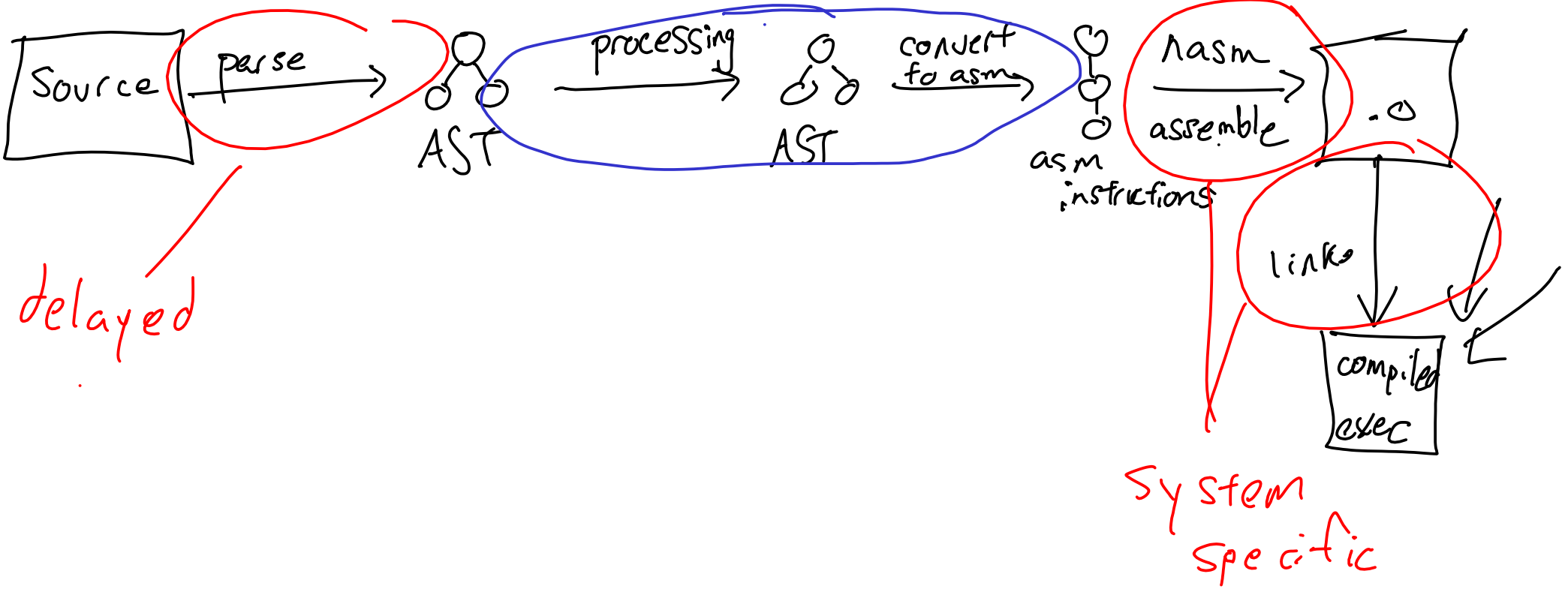
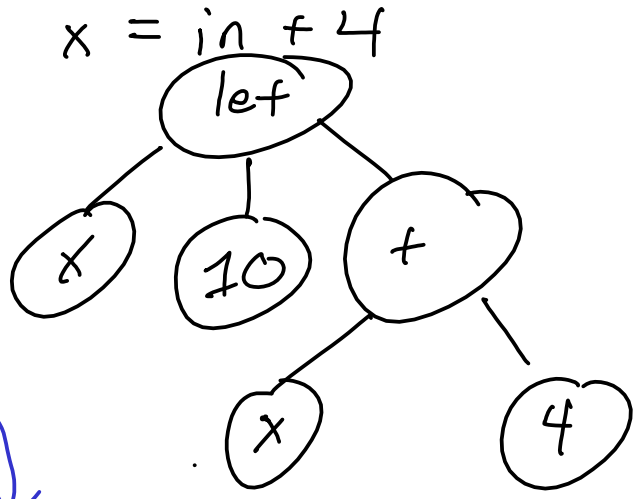

Source
parse
AST
processing
AST
convert to asm
asm instructions
nasm
assemble
.o
link
compiled exec
delayed
System specific

Parsing & ASTs



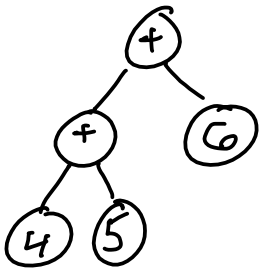
let x = 10 in x + 4

EBNF

<expr> ::= let <ident> = <expr> in <expr>
| <integer>
| <expr> + <expr>
| <ident>
| (<expr>)

concrete syntax

(4+5)+6

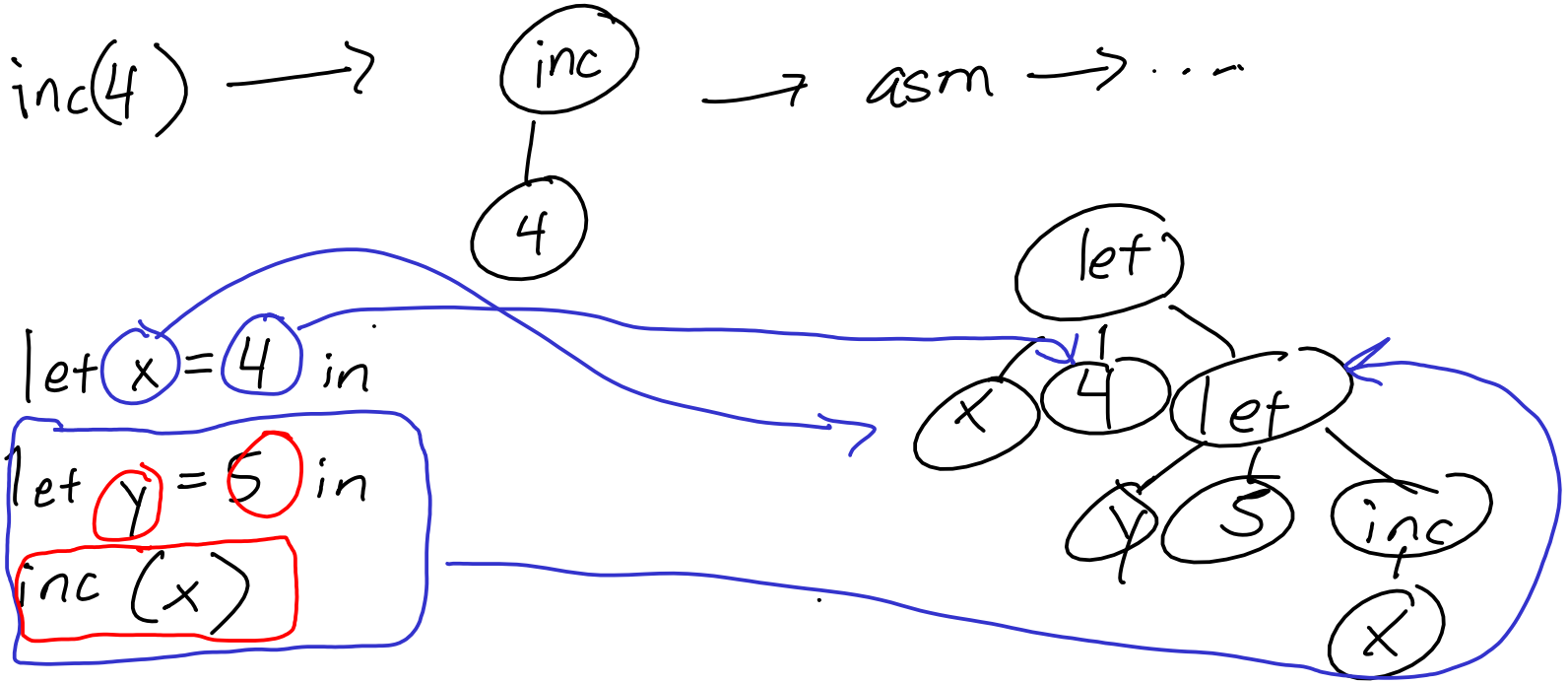


abstract syntax

```
type expr =
  |ELet of string * expr * expr
  |EInt of int
  |EPlus of expr * expr
  |EVar of string
```

Adder

<expr> ::= <integer>
| inc(<expr>)
| dec(<expr>)
| let <identifier> = <expr> in <expr>
| <identifier>
| (<expr>)



x86

Intel syntax

~~mov eax, $4~~

```
mov eax, 4
add eax, 1
```

```
[ XMov(
    XRegister(EAX),
    XConstant(4))
;
  XAdd(
    XRegister(EAX),
    XConstant(1))
]
```

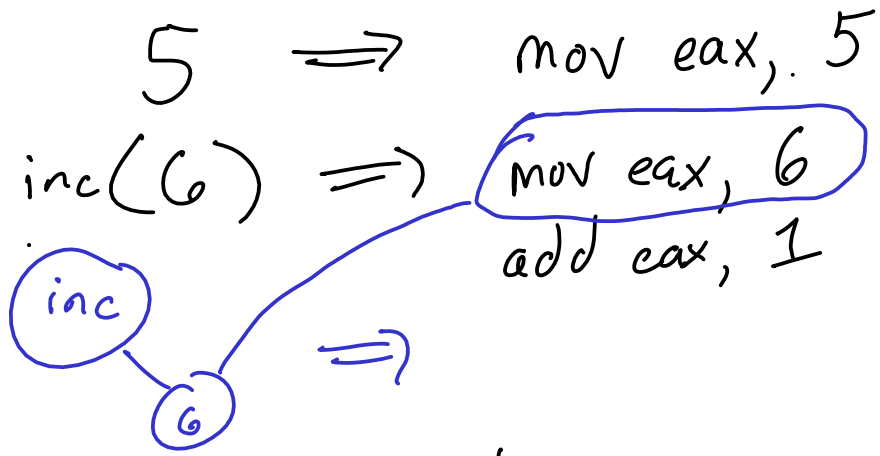
```
let compile (e : expr) : instruction list =
  ...
```

```
type arg =
  | XRegister of register

type instruction =
  | XMov of arg * arg
```

Adder Compilation

eax : return values




```
let rec compile (e : expr) : instruction list =
    match e with
    | EInt n -> [XMov(XRegister EAX, XConstant n)]
    | EInc e2 ->
        let instr = compile e2 in
        instr @ [XAdd(XRegister EAX, XConstant 1)]
```