

Unary: convert all args
[except for Pascal var
params: just load the
local (per)]

Unary: $R \rightarrow R$
Must defer L-values before
Unary conversion

What about new & dispose
var
 $p : \wedge \text{Integer};$
:
new(p) 

new(expr)
L-value of pointer type
1. push(expr) on stack
(evaluate it, don't defer)
2. 
a) malloc(), to malloc().
b) push the size of the
pointer object type
(Integer L-value example)
c) malloc(bytes("value"))
(pushes address of new location
onto stack)
3. malloc(...), heap()

Apr 6-11:04 AM

dispose(p)
dispose(expr)
R-val 
dispose(
):
handle_arg_list(
)
evaluate expr as R-value
(pointer to auto stack)
handle_arg_list("free", args)
b_free_all_params("free", args)

Enter credit local of Proj II:
function foo(Pascal): Real;
var
a, b: Integer;
begin
[foo]
: foo := 2.0;
a := 2.0;
end

Installing in a local block
requires knowledge of the offset
b: local_block_offset()
for each formal param
left to right:
a: b: local_block_offset()
offset to type of the param
offset to var param, var PPR
If function, call
push_block_return_value()
param var section
(local var decls):

Apr 6-11:18 AM

to install a local var:
get the offset by calling
local_block_offset()
(before passing args)

local_block_offset() (co)
local_block_offset()
In compute offsets yourself based
on types of local vars declared
alignment requirements is same as
size.
Remember the total space needed
for local var

How to determine if var decls are
local? If Stackable() > 0
check offset of var
var decls finished, get to
body of foo

before passing body:
local_block_offset() (function)
For each formal param,
call local_block_offset() &
of param, var param
over stack at offset computed
center to local_block_offset()
offset is returned
local_block_offset() (this should be
the same value per offset)
local_block_offset()
[don't do it before!
my mistake]
instead, get offset of local var
if var type is local var

Apr 6-11:32 AM


local_block_offset() (size)
var start passing statements
foo := expr
local_block_offset() // if
when done:
local_block_offset() // if
local_block_offset()

Project III
- control flow
- array references
structural statement:
compound stmt // loop body
begin - end
conditional
if x < 3 then
statement
endif
Backward:
label (string)
non-symbolic // good for
jump (label) // unconditional
jump to label
bcond_jump (condition, label)
bcond_jump

Apr 6-11:50 AM

simple_if:

test
LEX IF boolean expression LEX THEN statement

text section
of assembly code
"cont"
{ \$B = new_symbol();
bcond_jump(B3, "cont"); }

code to eval
test
jump to
cont if
p < 0
code for
statement
cont:
{ label("cont");
\$4 }

Apr 6-12:07 PM