Algorithm Gen_Opt_Code

Algorithm	$RSTACK$-Contents resp. result register
var $RSTACK$: **stack of register**;	
var $TSTACK$: **stack of address**;	
proc $Gen\_Code\ (t:$ **tree**);	$(R', R', \ldots)$
var R: **register**, T: **address**;	
case t **of**	
$(leaf\ a, 1):$ (∗left leaf∗)	
$emit(top(RSTACK) := a)$;	result in R'
$op(t_1, r_1), (leaf\ a, 0))$: (∗right leaf∗)	
$Gen\_Code(t_1)$;	
$emit(top(RSTACK) :=$ $top(RSTACK)$ **Op** $a)$;	result in R'

$op(t_1, r_1), (t_2, r_2))$:	$(R', R'', \ldots)$
cases	
$r_1 <$ $min(r_2, r)$:	$(R', R'', \ldots)$
begin	
exchange$(RSTACK)$;	$(R'', R', \ldots)$
$Gen\_Code(t_2)$;	result in R''
$R :=$ **pop**$(RSTACK)$;	$(R', \ldots)$
$Gen\_Code(t_1)$;	result in R'
$emit(top(RSTACK)$ $:=$ $top(RSTACK)$ **Op** $R)$;	result in R'
push$(RSTACK, R)$;	$(R'', R', \ldots)$
exchange$(RSTACK)$;	$(R', R'', \ldots)$
end ;	

$r_1 \geq r_2 \wedge r_2 < r$:	$(R', R'', \ldots)$
begin	
$Gen\_Code(t_1)$;	result in R'
$R :=$ **pop**$(RSTACK)$;	$(R'', \ldots)$
$Gen\_Code(t_2)$;	result in R''
$emit(R := R$ **Op** $top(RSTACK))$;	result in R'
push$(RSTACK, R)$;	$(R', R'', \ldots)$
end ;	

$r_1 \geq r \wedge r_2 \geq r$:	$(R', R'', \ldots)$
begin	
$Gen\_Code(t_2)$;	result in R'
$T :=$ **pop**$(TSTACK)$;	
$emit(M[T] := top(RSTACK))$;	result in $M[T]$
$Gen\_Code(t_1)$;	result in R'
$emit(top(RSTACK) := top(RSTACK)$ **Op** $M[T])$;	result in R'
push$(TSTACK, T)$;	
end ;	
endcases	
endcase	
endproc	