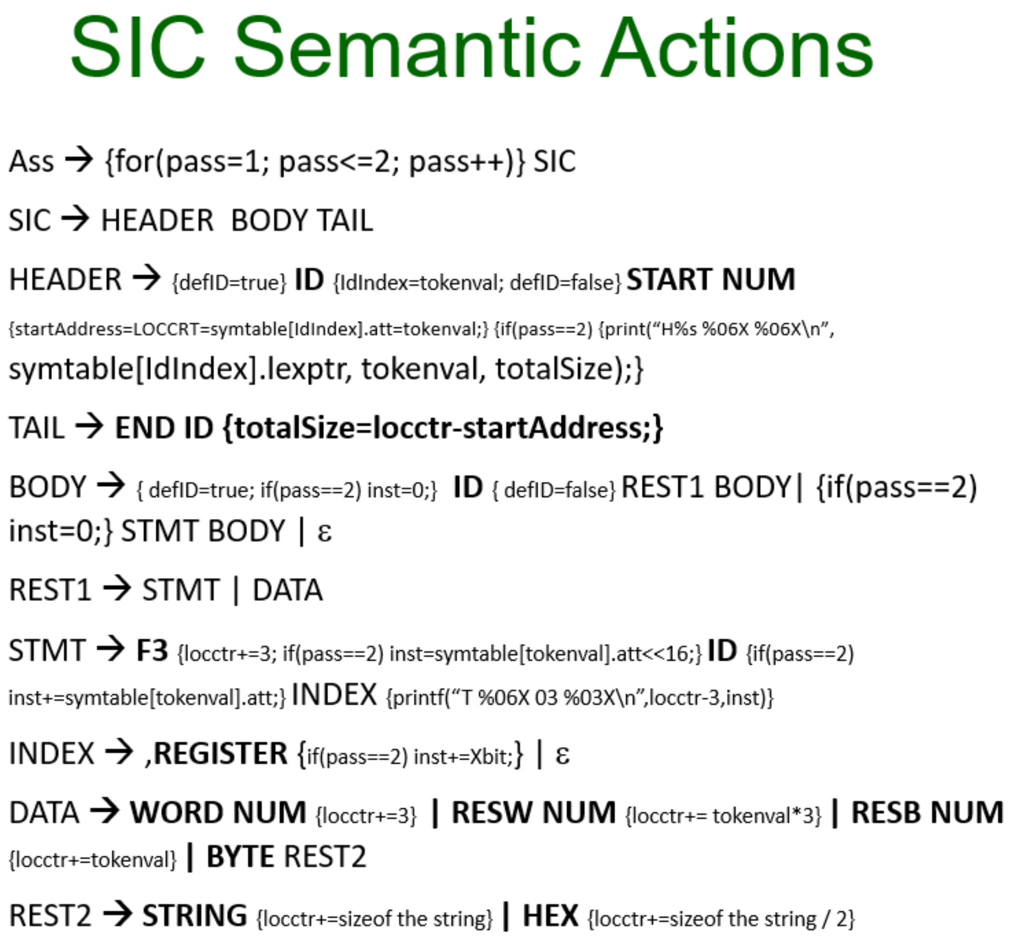

SIC Semantic Actions

Ass → {for(pass=1; pass<=2; pass++)} SIC

SIC → HEADER BODY TAIL

HEADER → {defID=true} **ID** {IdIndex=tokenval; defID=false} **START NUM** {startAddress=LOCCRT=symtable[IdIndex].att=tokenval;} {if(pass==2) {print("H%s %06X %06X\n", symtable[IdIndex].lexptr, tokenval, totalSize);}

TAIL → **END ID {totalSize=locctr-startAddress;}**

BODY → { defID=true; if(pass==2) inst=0;} **ID** { defID=false} REST1 BODY| {if(pass==2) inst=0;} STMT BODY | ε

REST1 → STMT | DATA

STMT → **F3** {locctr+=3; if(pass==2) inst=symtable[tokenval].att<<16;} **ID** {if(pass==2) inst+=symtable[tokenval].att;} INDEX {printf("T %06X 03 %03X\n",locctr-3,inst)}

INDEX → **,REGISTER** {if(pass==2) inst+=Xbit;} | ε

DATA → **WORD NUM** {locctr+=3} **| RESW NUM** {locctr+= tokenval*3} **| RESB NUM** {locctr+=tokenval} **| BYTE** REST2

REST2 → **STRING** {locctr+=sizeof the string} **| HEX** {locctr+=sizeof the string / 2}

For SICxe:
only change stmt and rest2, and add rest4 & rest5
stmt -> F1 | F2 Reg Rest5
| F3 rest2
| +F3 rest2 (you will use a boolean)

rest5-> , (coma) reg | epsilon (the other reg or nothing)
rest2-> ID Index | Num | @ rest4 | # rest4 | epsilon
rest4 -> Id | num