Zadatak 1. (8 poena) Definisati 8086 kod i izgled aktivacionog sloga za funkciju verify_preconditions. Za predstavljanje podataka tipa int kao i memorijskih adresa koriste se 2 bajta. Pretpostaviti da se rezultat funkcije verify_preconditions smešta na stek, a rezultat funkcije

```
registar
exec query
struct verification rule
                               int verify preconditions (
                                 struct verification rule* vr) {
                                 int truth=0;
  int type;
  struct verification rule
                                 int condition=1;
    *next;
                                 if (vr->type == 0) {
 int query id;
                                   condition = condition *
                                     exec query(vr->query id);
                                 if(vr->next != 0)
                                   condition *=
                                     verify_preconditions(vr->next);
                                  return condition;
```

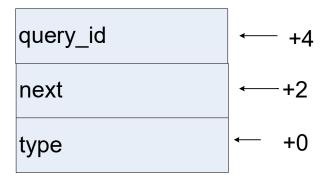
Komentar: Sve promenljive int tretiramo da su 16bit, a podrazumevamo da se kontekst procesora čuva na početku, a restaurira na kraju potprograma. Zbog toga su ovi delovi u rešenju izostavljeni. Osim toga, interesuje nas samo 16bit rezultat množenja iz AX, po dogovoru.

Napomena: Moguće je da u rešenju postoje greške, tako da su sugestije na mail dobrodošle. Osim toga, pojedine optimizacije prilikom generisanja koda u ovom rešenju nisu primenjene.

Aktivacioni slog



Struct verification rule



8086 kod:

upis BP na stek

PUSH BP

promena vrednosti pokazivaca BP:

MOV BP, SP

Rezervacija prostora za lokalne promenljive truth i condition, 4 jer je 2*2byte int:

SUB SP, 4

Telo funkcije:

MOV [BP-2], 0 MOV [BP-4], 1

Pribavi *vr

MOV BX, [BP+4]

Pribavi vr->type:

MOV AX, [BX] CMP AX,0 JNE lab1

MOV DX, [BX+4]

Poziv exec_query:

PUSH AX

CALL EXEC_QUERY

ADD SP, 2

Uzimanje rezultata iz CX:

MOV DX, CX

Množenje:

AX, [BP-4] VOM AX, DX MUL [BP-4], AX VOM lab1 MOV BX, [BP+4] CMP [BX+2],0JΕ kraj MOV AX, [BP-4] VOM DX, [BX+2]

Rekurzivni poziv:

Alokacija mesta za rezultat:

PUSH AX

Alokacija mesta za argument vr->next:

PUSH DX

CALL VERIFY PRECONDITIONS

Osloboditi argument:

ADD SP, 2

```
Uzeti rezultat sa steka:
```

POP AX

Pribaviti condition:

MOV DX, [BP-4]

Množiti sa rezultatom rekurzivnog poziva

MUL AX, DX

Vratiti u condition:

MOV [BP-4], AX

Upis rezultata na vrh steka:

kraj MOV AX, [BP-4]

MOV [BP+6],AX

Izbacivanje lokalnih promenljivih iz steka:

MOV SP, BP

Uzimasnje stare vrednosti BP-a:

POP BP

RET