Luentomonistetta ”Tietokoneen toiminta, Auvo Häkkinen, 1998” mukaillen.

Monipuolinen esimerkki

Lopuksi monipuolinen esimerkki, jossa ohjelmisto koostuu pääohjelmasta ja kahdesta aliohjelmasta, joista toinen on proseduuri ja toinen funktio.

Funktio Sqrt(int n):

```
int Sqrt(int n)  {

  return n * n;

}
```

Proseduuri Summa(int &X, int Y):

```
void Summa(int &X, int Y)  {

  int Z;

  Z = A + B;
  X = Sqr(X) + Y + Z + 5;

}
```

Erohan näillä on se, että funktiolla on paluuarvo, ja proseduurilla ei ole. A ja B ovat pääohjelman globaaleja muuttujia.

Itse pääohjelma pseudokoodina näyttäisi seuraavalta:

```
int A, B;

lue luku A;
B = Sqr(2);
tulosta B;
Summa (B, A);
tulosta B;
```

Symbolisena konekielenä nämä ohjelmat näyttäisivät seuraavilta:

Luentomonistetta ”Tietokoneen toiminta, Auvo Häkkinen, 1998” mukaillen.

Pääohjelma:

	A DS 1	Alustetaan ”minitaulukko” A (”A DS 1” luo yhden alkion kokoisen taulukon).
	B DS 1	Ja alustetaan ”minitaulukko” B.
Main	**IN R3, =KBD**	Luetaan näppäimistöltä arvo rekisteriin R3...
	STORE R3, A	...ja tallennetaan se muuttujaan A (tai jos ihan tarkkoja ollaan, ”taulukko” A:n ensimmäisen, ja ainoan, alkion arvoksi).
	PUSH SP, =0	Varataan paluuarvolle paikka.
	PUSH SP, =2	Pushataan vakio 2 pinoon (halusimmehan laskea kahden toisen potenssin Sqr-funktiolla).
	CALL SP, Sqr	Kutsutaan Sqr-funktiota.
	POP SP, R2	Sqr funktio talletti paluuarvolle varattuun paikkaan laskennan tuloksen, joten popataan se nyt rekisteriin R2.
	STORE R2, B	Ja talletetaan funktion palauttama arvo muuttujaan B.
	OUT R2, =CRT	Ja tulostetaan se vielä näytölle. Koska B:n arvo on jo rekisterissä R2, sitä olisi ihan turha ensin ladata rekisteriin LOAD-käskyllä.
	PUSH SP, =B	Laitetaan B:n osoite pinoon (viiteparametri proseduurille Summa).
	PUSH SP, A	Laitetaan A:n arvo pinoon (arvoparametri proseduurille Summa).
	CALL SP, Summa	Kutsutaan Summa:a.
	LOAD R2, B	Proseduuri Summa laittoi laskentansa tuloksen suoraan muuttujaan B, joten mitään ei tarvitse popata (eikä voidakaan, koska Summa ei pushannut mitään pinoon), ladataan vain muuttujan arvo suoraan rekisteriin.
	OUT R2, =CRT	Ja tulostetaan B:n arvo näytölle.
	SVC SP, =HALT	Loppu.

Luentomonistetta ”Tietokoneen toiminta, Auvo Häkkinen, 1998” mukaillen.

Aliohjelmat:

Funktio Sqr:

	n EQU -2	Kuten edellisistä esimerkeistä (yksinkertainen aliohjelma ja parametrit aliohjelmassa) varmaan muistat, pinossa välitetyt parametrit löydetään suhteessa FP:n arvoon. Ne ovat kohdassa FP-2 ja siitä ”taaksepäin”, riippuen kuinka monta parametria on. Paikat ”0” ja ”-1” olivat ne vanhan PC:n ja vanhan FP:n arvot.
	Sqr_Ret EQU -3	Paluuarvo on pinossa ennen parametrejä, joten sen paikan osoite on yhden pienempi kuin ensimmäisen parametrin. Koska tässä tapauksessa on vain yksi parametri kohdassa FP-2, paluuarvon paikka löytyy kohdasta FP-3.
Sqr	**LOAD R3, n(FP)**	Ladataan rekisteriin pinosta haettu ”n”...
	MUL R3, R3	...josta sitten lasketaan n * n.
	STORE R3, Sqr_Ret(FP)	Ja tämä talletetaan paluuarvon paikkaan pinossa.
	EXIT SP, =1	Eikä sitten muuta. Poistutaan aliohjelmasta ja potkaistaan samalla se välitetty parametri ulos pinosta siirtämällä pino-osoitinta, jotta ohjelma löytää oikean vanhan PC:n arvon palatessaan pääohjelmaan.

Luentomonistetta ”Tietokoneen toiminta, Auvo Häkkinen, 1998” mukaillen.

Proseduuri Summa:

	X EQU -3	Nimetään X:n paikka pinossa suhteessa FP:hen.
	Y EQU -2	Nimetään Y:n paikka pinossa suhteessa FP:hen.
	Z EQU 1	Nimetään Z:n paikka koodissa suhteessa FP:hen. Z on aliohjelman oma (paikallinen) muuttuja, siksi sen paikka on positiivinen. Teoriassa tätä ei olisi ollut pakko laittaa pinoon, oltaisiin voitu myös luoda muuttuja. Mutta käytetään nyt pinoa kun se on kuitenkin aliohjelma Eipähän se muuttuja jää sitten haahuilemaan pääohjelmatasolla. Kun on paikallinen niin pitäisi hyvä tyylin mukaan pysyä aliohjelman sisällä. ☺
Summa	**PUSH SP, =0**	Varataan muuttujalle Z tilaa pinosta. Se että sille luotiin edellisessä (pseudo)käskyssä ”paikka” ei vielä tehnyt pinolle mitään.
	LOAD R1, A	Ladataan A (pääohjelmatason muuttuja) rekisteriin R1.
	ADD R1, B	A = A + B (B on myös pääohjelmatason muuttuja).
	STORE R1, Z(FP)	Talletetaan saatu arvo Z:aan (eli annetaan osoitteeksi Z:n paikka FP rekisterin suhteen), jolloin saadaan Z = A + B.
	PUSH SP, =0	Valmistaudutaan kutsumaan funktiota Sqr laittamalla sen paluuarvolle paikka pinoon.
	PUSH SP, @X(FP)	Tämä onkin hivenen monimutkaisempi. Sqr-funktiollehan piti antaa arvoparametri ”n”, jonka neliön se laskee. Haluamme antaa sille X:n arvon, mutta Summa sai vain X:n osoitteen (viiteparametri). Ei hätää, koska tiedämme X:n osoitteen, käytämme vain kaksinkertaista muistinoutoa, jolla saamme X:n arvon.
	CALL SP, Sqr	Kutsutaan Sqr-funktiota.
	POP SP, R3	Sqr laittoi paluuarvonsa pinoon, joten popataan se pois.

Luentomonistetta ”Tietokoneen toiminta, Auvo Häkkinen, 1998” mukaillen.

	ADD R3, Y(FP)	Lisätään Sqr-funktion palauttamaan arvon Y:n arvo.
	ADD R3, =5	Ja lisätään edelliseen summaan vielä 5. Jolloin rekisterissä R3 on summa Sqr(5) + Y + 5.
	ADD R3, Z(FP)	Ja lisätään vielä Z:n arvo, jolloin rekisterissä on Sqr(5) + Y + Z + 5.
	STORE R3, @X(FP)	Ja sitten talletetaan koko summa viiteparametrina annettuun osoitteeseen, eli tässä tapauksessa muuttujan B arvoksi (koska pääohjelmatasolla Summaa kutsuttiin parametreilla (B, A)).
	SUB SP, =1	Pudotetaan paikallinen muuttuja pois pinosta. Koska niitä oli vain yksi, vähennetään pino-osoittimen arvosta yksi. Jos niitä olisi ollut kaksi, vähennettäisiin kaksi, jne.
	EXIT SP, =2	Ja poistutaan aliohjelmasta, taas antaen toiseksi operandiksi aliohjelmalle annettujen parametrien määrän, että ne voidaan poistaa pinosta.