Treća laboratorijska vježba iz 'Arhitektura računala 1'

Vježba 5

Zadatak ove laboratorijske vježbe je napisati program za procesor ARM7 koji će rješenja jednadžbe {$Y=(X^3-1)/(2*X)$} pohranjivati od adrese 400_{16}, a podatak 'X' uzima od adrese 100_{16} to svu redom podaci $\{0, 3, 6, -1, -6\}_{16}$ i terminirani su brojem '80000000_{16}' . Rješenja za provjeru ispravnosti su redom $\{00000000, 00000004, 00000011, 00000001, 00000012\}_{16}$. Potrebno je isprogramirati potprograme 'KUB' i 'DIV' u koje se parametri prenose stogom, a vraćaju registrom, koji čuvaju i obnavljaju kontekst.

Krenimo redom po kodu objasniti liniju po liniju.

Prvo ćemo inicijalizirati sve registre koje ćemo koristiti. Budući da koristimo potprograme, odnosno njima šaljemo argumente pomoću stoga. Dakle trebamo inicijalizirati SP (vrh stoga)= R13, koji se najčešće inicijalizira na '8000_{16}'. Nadalje spremamo u odabrane registre adrese podataka te odredišnu adresu. U R5 se postavlja konstatna '2_{16}' za kasnije udvostručenje podatka X jer naredba MUL (mulply) koristi isključivo registarsko množenje, ne možemo pomnožiti registar s konstantom.

Ulazimo u glavni dio koda označen labelom 'GLAVNI'. U glavnom djelu koda učitavamo podatke sa adrese na kojoj se ulazni podaci nalaze. Koristimo postindeksiranje 'LDR R1, [R0], #4', da bi nam se adresa R0, prilikom svakog prolaza, uvečala za 4, tj. da bi se u iducoj iteraciji mogao procitati iduci podatak. Provjeravamo jel broj jednak '80000000_{16}' jer on terminira niz ulaznih podataka. Ako je uvijet ispunjen, došli smo do kraja niza i skačemo na labelu 'KRAJ' na kojoj se nalazi naredba 'SWI 123456' (software interupt), iznimka, koja zaustavi procesor poput naredbe 'HALT' u FRISC-u. Međutim, ako uvijet nije ispunjen te je učitan jedan od ulaznih podataka nastavljamo dalje s kodom. Dolazimo do množenja učitanog podatka s konstantom u registru R5 te nam je to drugi argument u potprogramu 'DIV'.

Stavimo na stog učitani podatak 'X' koji se zbog načina pohrane na stog 'STMFD (Store Multiple Full Descending)' pohrani na adresu za '4_{16}' manju od inicijalne vrijednosti SP-a, odnosno na adresu '$7FFC_{16}$' što znači da se u tom naćinu adrese smanjuju od inicijalen prema '0_{16}'. Naredba 'BL KUB' je istovjetna 'CALL KUB' is FRISC-a, samo što se povratna adresa ne sprema na stog već u registar R14.

U potprogramu 'KUB' prvo stavljamo na stog registre kojima čemo mijenjati vrijednosti, učitamo naredbom LDR sa adrese SP-a s pomakom od '4_{16}' podatak 'X' koji ćemo kubirati te inicijaliziramo registar preko kojeg vraćamo rezultat potprograma. Samo kubiranje se izvodi dvostrukim množenjem broja sa samim sobom. U prvom množenju se rezultat 'X^2' pohrani u registar za vraćanje rezultata te u drugom množenju rezultat 'X^2' * 'X' također pohranimo u registar za vraćanje rezultata.

Nakon povratka, u glavni dio koda, nam preostaje počistiti stog što radimo naredbom koja uvećava SP (registar R13) na inicijalnu vrijednost. Budući da smo poslali jedna podatak na stog prije poziva potprograma moramo SP uvećati za '4_{16}' da bi bio vraćen na inicijalno vrijednost. Nakon toga nam preostaje *finiširati* prvi argument potprograma 'DIV', a to je oduzeti '1_{16}' od vraćenog skubiranog podatka.

Oba argmenta R3=(X^3-1) i R2=(2*X) šaljemo na stog te pozivamo potprogram 'DIV' u kojem ponovno prvo na stog stavljamo registre s kojima ćemo nešto petljati. Moramo inicijalizirati registar za vraćanje rezultata djeljenja. Nadalje učitavamo pomoću naredbe 'LDR' i offseta SP-a naše prethodno stavljene na stog argumente. Idući korak je provjera: "jesu li brojevi negativni?", budući da radimo sa 2'k brojevima, a za djeljenje kao uzastopno oduzimanje trebamo isključivo pozitivne brojeve (repetitivno oduzimanje djelitelja od djeljenika. R3-R2). Ako su negativni koristimo naredbu 'RSB Rn, Rn, #0', za n={2,3} = (0-Rn), za Rn<0 imamo "0-(-Rn)' = (0+Rn) = Rn dakle dobijemo pozitivan broj u Rn iz negativnog 2'k broja naredbom 'RSB (Reverse SuBstract)'. Sljedeći korak je uzastopno oduzimati R2 od R3 (R3-R2) I povećava registar reultata svakom iteracijom za '1_{16}' do kad se ne zadovolji uvjet R3<R2 te se u tom trenu izlazi iz potprograma. Još jedan uvijet treba provjeriti, a to je da djelitelj (R2) nije '0_{16}' jer bi to značilo beskonačno djeljenje s R2 što bi bila beskonačna petlja, a to želimo izbjeći. U slučaju da je uvijet zadovoljen, također se izlazi iz potprograma te je povratna vrijednost nula.

Ponovnim povratkom u glavni program preostaje nam isprazniti stog na koji smo stavili argumente svog potprograma. Ovaj puta smo slali dva argumenta, a to znači da moramo registru R13 dodati vrijednost '8_{16}' da bi se on (SP) vratio u inicijanu vrijednost. Konačno imamo željeni rezultat funkcije spremljen u registru R7 (prenesenom iz potprograma 'DIV') te ga moramo spremiti na unaprijed zadanu adresu rezultata, al na način 'STR R7, [R4], #4', da bi se registar R4, koji nam čuva vrijednost adrese pohrane rezultata zadane funkcije, uvećao naknadno za '4_{16}'. Na taj način (postindeksiranjem) će u idućoj iteraciji u R4 biti pohranjena adresa '404_{16}'. Zadnja naredba glavnog dijela koda nas vraća na početak glavnog dijela kako bi ralunali vrijednost funkcije 'Y' za idući ulazni podatak.

Vježba 6

Bit će možda jednog lijepog sunčanog posljepodneva pretočena iz usmene predaje na virutalni papir.