**Ljetni ispitni rok iz ARH 1 2. srpnja 2012. Grupa na predavanjima: \_\_\_\_\_\_\_\_**

**Prezime i ime (velikim slovima):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ JMBAG: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Izjavljujem da tijekom izrade ove zadaće neću od drugoga primiti niti drugome pružiti pomoć, te da se neću koristiti nedopuštenim sredstvima. Ove su radnje teška povreda Kodeksa ponašanja te mogu uzrokovati i trajno isključenje s Fakulteta. Izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.**

**Potpis:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.**

**Dozvoljeno je koristiti isključivo službene šalabahtere (popis naredaba FRISC-a i ARM-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadateke rješavati na ovaj papir. Ispit traje 150 minuta.**

**1.** (*1 bod*)

Za procesor ARM napišite programski odsječak koji izvodi predznačno proširenje broja u registru R0 iz 8-bitnog zapisa 2'k, na 32 bita. Upotrijebite dvije naredbe MOV.

**MOV R0, R0, LSL #18**

**MOV R0, R0, ASR #18**

**2.** (*1 bod*)

Za sljedeće neposredne vrijednosti navedite je li ih moguće upisati kao dio naredbe procesora ARM.

0x02080000 \_\_\_\_\_\_\_\_\_\_ 0x0000FFFF \_\_\_\_\_\_\_\_\_\_

0xFFFF34FF \_\_\_\_\_\_\_\_\_\_ 0x00000101 \_\_\_\_\_\_\_\_\_\_

**3.** (*0.5 boda*)

Kod procesora FRISC, uvjetno izvođenje naredbi moguće je za **UPRAVLJAČKE** naredbe.

Kod procesora ARM, uvjetno izvođenje naredbi moguće je za **SKORO SVE** naredbe.

**4.** (*2 boda*)

Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe JP\_NV 500:

Razina dohvata: Razina izvođenja:

Rastući brid CLOCK-a: Rastući brid CLOCK-a: **-**

\_\_\_\_\_**PC 🡪 AR**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Padajući brid CLOCK-a:

Padajući brid CLOCK-a: \_**ako V=0: ext 500 🡪 PC**

\_\_\_\_\_**PC+4 🡪 PC**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ \_**omogući dohvat u sljedećem ciklusu**

\_\_\_\_\_**(AR) 🡪 IR, dekodiranje\_\_\_\_\_\_\_\_\_\_\_**

\_\_\_\_\_**ispitivanje istinitosti uvjeta V=0**\_\_\_\_

\_\_\_\_\_**ako V=0: ext 500**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_**onemogući dohvat u sljedećem ciklusu**

**5.** (*1 bod)*

S obzirom na smještaj operanada postoje procesorske ahitekture: **stogovna**, **akumulatorska, registar – memorija, registar-registar.**

Za RISC procesore, od ove 4 arhitekture uobičajeno se koristi **registar-registar** .

**6.** *(1 bod)*

Procesor ARM izvodi sljedeću instrukciju LDM R13, {R1} , pri čemu su početni sadržaji registra R13=10016 i R1=0102030416 . Sadržaj memorijskih lokacija prikazan je na slici. Odredite sadržaje registara R13 i R1 nakon izvođenja instrukcije.

|  |  |  |  |
| --- | --- | --- | --- |
| **Adresa** | **Podatak** | **Adresa** | **Podatak** |
| 103 | 04 | 0FE | 09 |
| 103 | 05 | 0FD | 0A |
| 101 | 06 | 0FC | 0B |
| 100 | 07 | 0FB | 0C |
| 0FF | 08 | 0FA | 0D |

R13=\_\_\_\_\_\_**100**\_\_\_\_\_\_\_\_\_  
R1=\_\_\_**0708090A**\_\_\_\_\_\_\_\_

**7.** *(1 bod)*

Dopunite stanje signala i sadržaje registara za sljedeće korake kod prihvaćanja INT3 na FRISC-u.

IACK ← \_**1**\_\_

IIF ← \_**0**\_\_

\_**SP**\_ ← PC

IACK ← \_**0** \_

PC ← \_**12(dek)**\_\_

**1. FRISC** (*8 bodova*)

Za procesor FRISC napišite program koji računa skalarni produkt dva vektora. Vektori su smješteni u memoriji kao blokovi   
32-bitnih NBC podataka (npr. vektor **a**=[a0, a1, a2, ... , an] i vektor **b**=[b0, b1, b2, ... , bn]. Skalarni produkt dva vektora računa se po formuli: SP (a,b) =

Adresa bloka prvog vektora nalazi se na adresi 100016, a adresa bloka drugog vektora nalazi se na adresi 200016. Duljina (dimenzija) vektora nalazi se na lokaciji DULJINA.

Napišite potprogram PRODUKT za računanje skalarnog produkta dva vektora. Adresa prvog podatka vektora **a** nalazi se u registru R0, a adresa prvog podatka vektora **b** u registru R1. Vektori su iste duljine, koja se u potprogram prenosi preko registra R2. Rezultat treba vratiti preko memorijske lokacije REZ. Pretpostavite da prilikom zbrajanja neće doći do prekoračenja opsega.

Napišite potprogram MNOZI za množenje dva 32-bitna NBC broja metodom uzastopnog zbrajanja (pretpostavite da neće doći do prekoračenja opsega). Ulazni podaci prenose se preko stoga, a rezultat treba vratiti u registru R0. Parametre uklanja pozivatelj.

**̀ORG 0**

**MOVE 10000, SP**

**MOVE 1000, R0**

**MOVE 2000, R1**

**LOAD R2, (DULJINA)**

**CALL PRODUKT**

**HALT**

**PRODUKT:**

**MOVE R0, R3**

**MOVE 0, R4**

**PETLJA PUSH R0**

**PUSH R1**

**CALL MNOZI**

**ADD SP, 8, SP**

**ADD R0, R4, R4**

**ADD R0, 4, R0**

**ADD R1, 4, R1**

**SUB R2, 1, R2**

**JR\_NZ PETLJA**

**STORE R4, (REZ)**

**RET**

**MNOZI:**

**PUSH R1**

**PUSH R2**

**LOAD R1, (SP+10)**

**LOAD R2, (SP+C)**

**MOVE 0, R0**

**LOOP ADD R1, R0, R0**

**SUB R2, 1, R2**

**JR\_NZ LOOP**

**POP R2**

**POP R1**

**RET**

**DULJINA DW ...**

**REZ DW 0**

**2. FRISC** (*9 bodova*)

U računalnom sustavu nalazi se procesor FRISC, sklop PIO (adresa 0FFFF1000), prekidna vanjska jedinica PVJ (adresa 0FFFF2000, spojena na INT0) na koju je spojen motor, te sklop CT (adresa 0FFFF3000, spojen na INT1).

Prekidna jedinica PVJ mjeri brzinu vrtnje nekog motora i generira prekid svaki put kada motor napravi puni okret. Sklop PIO služi za upravljanje motorom i radi u bezuvjetnom načinu rada. Slanje vrijednosti 0 na PIO isključuje motor, a slanje vrijednosti 1 uključuje motor.

Svake sekunde potrebno je mjeriti brzinu vrtnje motora. Ako je ona veća od 50 okretaja/sekundi motor treba isključiti, pričekati 5 sekundi, te ponovo uključiti. Nakon uključivanja motora potrebno je ponovo mjeriti brzinu vrtnje. Postupak se beskonačno ponavlja. Na CT je spojen signal frekvencije 1 kHz.

**PIOCR ̀EQU 0FFFF1000**

**PIODR ̀EQU 0FFFF1004**

**CTLR ̀EQU 0FFFF3000**

**CTCR ̀EQU 0FFFF3004**

**CTSR ̀EQU 0FFFF3008**

**CTEND ̀EQU 0FFFF300C**

**PVJPOD ̀EQU 0FFFF2000**

**PVJTEST ̀EQU 0FFFF2004**

**PVJEND ̀EQU 0FFFF2008**

**̀ORG 0**

**MOVE 10000, SP**

**JP GLAVNI**

**̀ORG 8**

**DW 800**

**GLAVNI MOVE %B 100, R0**

**STORE RO, (PIOCR)**

**MOVE 1, R0**

**STORE RO, (PIODR)**

**MOVE %D 1000, R0**

**STORE RO, (CTLR)**

**MOVE %B 11, R0**

**STORE RO, (CTCR)**

**MOVE %B 10110000, SR**

**PETLJA JR PETLJA**

**̀ORG 800**

**PUSH R0**

**MOVE SR, R0**

**PUSH R0**

**LOAD RO, (CTSR)**

**CMP RO, 1**

**JR\_EQ CT\_PREKID**

**STORE R0, (PVJTEST)**

**LOAD R0, (POMOCNI)**

**ADD RO, 1, RO**

**STORE R0, (POMOCNI)**

**STORE R0, (PVJEND)**

**JR KRAJ**

**CT\_PREKID STORE RO, (CTSR)**

**LOAD R0, (POMOCNI)**

**CMP R0, %D 50**

**JR\_ULE KRAJ**

**MOVE 0, RO**

**STORE RO, (PIODR)**

**LOAD RO, (KONSTANTA)**

**LOOP SUB R0, 1, R0**

**JR\_NZ LOOP**

**MOVE 1, R0**

**STORE RO, (PIODR)**

**STORE RO, (CTEND)**

**KRAJ POP RO**

**MOVE RO, SR**

**POP R0**

**RETI**

**POMOCNI DW 0**

**KONSTANTA DW 1000**

**3. ARM** (*10 bodova*)

U memoriji procesora ARM nalazi se popis od 100016 16-bitnih podataka o artiklima prodanim u trgovini. Svaki zapisani podatak predstavlja ID-broj artikla (broj između 0 i 204710) koji je prodan.

Napisati potprogram BROJI koji će prebrojati koliko je komada pojedinog artikla prodano i napraviti izvještaj o broju prodanih komada artikla, poredan po ID-broju. Potprogram preko stoga prima dva ulazna parametra: adresu početka popisa prodanih artikla i adresu izvještaja. Potprogram u izvještaj zapisuje brojeve prodanih komada svakog artikla kao 16-bitne podatke. Na početnoj adresi tog izvještaja treba se nalaziti broj prodanih artikala s ID-brojem 0, iza tog podatka broj prodanih artikala s   
ID-brojem 1, itd. Nakon izrade izvještaja, potrebno je odrediti najprodavaniji artikl, pozivom potrprograma NAJVIŠE. ID-broj najprodavanijeg artikla treba spremiti u memorijsku lokaciju HIT.

Napisati potprogram NAJVIŠE koji treba odrediti koji artikl se najviše prodavao. Parametar adrese izvještaja predaje se potprogramu preko lokacije iza poziva potprograma. ID-broj najprodavanijeg treba vratiti preko registra R0.

Napisati glavni program koji će za ulazni blok koji počinje o adrese 100016, potprogramom BROJI napraviti izvještaj o broju prodanih komada artikala od adrese 700016.

̀**ORG 0**

**MOV SP, #10<12**

**MOV R0, #10<8**

**MOV R1, #70<8**

**STMFD SP!, {RO, R1}**

**BL BROJI**

**HALT**

**BROJI STMFD SP!, {R0, R1, R2, R3, R4, R14}**

**LDR R0, [SP, #1C]**

**LDR R1, [SP, #20]**

**MOV R2, #10<8**

**PETLJA LDRH R3, [R0], #2**

**LDRH R4, [R1, R3, LSL #1]**

**ADD R4, R4, #1**

**STRH R4, [R1, R3, LSL #1]**

**SUB R2, R2, #1**

**BNE PETLJA**

**BL NAJVIŠE**

**DW 7000**

**STR R0, HIT**

**STMFD SP!, {R0, R1, R2, R3, R4, R14}**

**MOV PC, LR**

**NAJVISE STMFD SP!, {R1, R2, R3, R4}**

**LDR R1, [R14], #4**

**LDR R2, KOLIKO**

**MOV R4, #0**

**MOV R0, #0**

**LOOP LDRH R3, [R1]**

**CMP R3, R4**

**MOVGT R4, R3**

**MOVGT R0, R1**

**ADD R1, R1, #2**

**SUB R2, R2, #1**

**BNE LOOP**

**LDMFD SP!, {R1, R2, R3, R4}**

**MOV PC, LR**

**KOLIKO DW 7FF**

**HIT DW 0**

**4. ARM** (*5 bodova*)

U računalnom sustavu nalazi se procesor ARM te sklopovi GPIO i RTC. Na vrata B sklopa GPIO spojeno je 8 senzora koji čine alarmni sustav. Napišite program koji svakih 10 sekundi obavlja provjeru stanja senzora. Ako je bilo koji od senzora postavljen u logičko “0“, treba aktivirati alarm upisom “1“ na bit 0 vrata A sklopa GPIO. Ako su svi senzori postavljeni u logičko “1“, treba deaktivirati alarm upisom “0“ na bit 0 vrata A sklopa GPIO. Adresa GPIO sklopa je FFFFF10016, a RTC sklopa FFFFE10016 . Na ulaz RTC sklopa spojen je signal frekvencije 1 kHz. Sklop RTC spojen je na IRQ.

̀**ORG 0**

**B GLAVNI**

**̀ORG 18**

**B 800**

**GLAVNI MOV SP, #10<12**

**LDR R0, RTC**

**LDR R1, GPIO**

**MOV R2, #0FF**

**STR R2, [R1, #8]**

**STR R2, [R0, #C]**

**MOV R2, #10<12**

**STR R2, [R0, #4]**

**MOV R2, #1**

**STR R2, [R0, #10]**

**MRS R2, CPSR**

**BIC R2, R2, #40**

**MSR CPSR\_c, R2**

**PETLJA B PETLJA**

**̀ORG 800**

**PREKIDNI STMFD SP!, {R0, R1, R2}**

**LDR R0, RTC**

**LDR R1, GPIO**

**LDR R2, [R1, #4]**

**AND R3, R2, #0FF**

**CMP R3, #0FF**

**MOVEQ R2, #0**

**STREQ R2, [R1]**

**MOVNE R2, #1**

**STRNE R2, [R1]**

**STR R2, [R0, #8]**

**MOV R2, #0**

**STR R2, [R0, #C]**

**LDMFD SP!, {RO, R1, R2}**

**SUBS PC, LR, #4**

**RTC DW 0FFFFE100**

**GPIO DW 0FFFFF100**