MICROCONTROLERE ȘI MICROSISTEME – APLICAȚII. Partea a III-a

HN Teodorescu

Iași 2016

Acest material complementează cele două volume anterioare apărute la editura UT Iași.

Materialul este destinat studenților de la disciplina de *Microcontrolere și microsisteme* (sau cursuri similare). Volumul are caracter pur didactic și include elemente teoretice și exerciții și întrebări propuse pentru a stimula înțelegerea și învățarea subiectului.

Materia efectiv predată de autor într-un anume an școlar nu include neapărat toate elementele din cele două volume și din prezentul material electronic.

Aceasta este o versiune provizorie, care poate include erori consistente; o versiune definitivă va apare ulterior.

Citarea recomandată pentru acest material: HN Teodorescu, Microcontrolere și microsisteme – aplicații. Partea a III-a. Iași 2016

Observații privind examinările

Candidații la examen iau la cunoștință că utilizarea sau deținerea asupra lor în timpul examenului a oricăror mijloace de stocare sau manipulare a informației, sau de comunicare nu este permisă și că, din momentul primirii subiectelor și până la părăsirea sălii de examen, nici o discuție cu colegii de examen nu este permisă.

Nerespectarea acestor condiții și/sau a procedurilor de examen conduce imediat la anularea necondiționată a tezei și a examenului respectivei/ului candidat(e).

Exercițiile de tip program se vor rezolva în **C şi asm**.

Toate exercițiile de la curs + proiectul sunt subiecte de examen !

Și subiectele privind partea de hard (curs) și cele privind partea de cod (lab.) sunt obligatorii și trebuie promovate cu minim 5.

Privind terminologia : registru – registre sau regiștri ?

Terminologia în l. română nu este bine fixată. Deși *Dexonline* dă[1] în diverse variante de explicații pentru cuvântul ‚registru' un singur plural, ‚registre', ca urmare a împrumutului din l. franceză, mulți autori români[2] (printre care și prezentul) preferă în cazul termenului tehnic pluralul ‚regiștri'. Alți autori care folosesc aceeași formă de plural sunt listați mai jos; autori care folosesc forma ‚registre' sunt exemplificați aici[3]. Acest autor preferă forma ‚regiștri' justificat și de apropierea mult mai mari a l. rom. de l. italiană, unde pluralul este ‚registri, (singular ‚registro'[4]), v. aici de ex.[5]

[1] https://dexonline.ro/definitie/registru

[2] http://forum.softpedia.com/topic/835079-registre-sau-regitri/, http://www.ibs.ro/~bela/Teachings/Microprocessors/LabDocs/Porturi.pdf, http://ep.etc.tuiasi.ro/site/Microcontrolere/Referate_laborator/lab3uC.pdf,

[3] http://ep.etc.tuiasi.ro/files/CID/registre.pdf , http://tet.pub.ro/files/studenti/materiale/an_III/miclab/lab1%20mC.pdf,

[4] De remarcat ca forma singular 'registru' este mai apropiată de 'registro' decât de cea de 'registr' conform pronuntiei franceze.

[5] http://www.microcontroller.it/Tutorials/PIC18/config18.htm, sau http://www.adrirobot.it/pathfinder/computer_microcontroller/pdf/CM_082.pdf

1. Puterea consumată și managementul puterii consumate în microprocesoare și microsisteme

1.1. Introducere

Pentru a reduce puterea consumată, microprocesoarele folosesc tehnologia porților CMOS. Sunt mai multe efecte de disipare a energiei pe poarta CMOS, unele dintre acestea dependente de frecvența de lucru, altele nu. Ca urmare, în expresia puterii disipate apar mai mulți termeni. Pierderile care depind de frecvența de lucru sunt numite pierderi dinamice; pierderile statice rămân constante cu frecvența. Uzual, pierderile dinamice sunt mult mai mari decât cele statice; ultimele sunt decisive în cazul regimului de lucru de hibernare (*sleep*).

Pentru proiectantul și fabricantul de microcontrolere și microprocesoare, proiectarea și realizarea trebuie să țină cont de influența tehnologiei asupra consumului de putere și ca urmare asupra aplicațiilor respectivului circuit. De regulă, cu cât aria chip-ului este mai mică, cu atât puterea consumată, la o frecvență dată este mai mică. Tranzistoarele de dimensiuni mai mari au tendința de a absorbi un curent mai mare. Aria mai mare a grilei face ca sarcina de comandă stocată pe grilă să fie mai mare, de unde și puterea consumată pentru încărcarea și descărcarea condensatorilor de grilă este mai mare, v. mai jos.

Proiectantul de aplicații, în special de aplicații mobile, pentru care consumul de putere este critic, trebuie să țină cont de influența frecvenței asupra pierderilor și să aleagă frecvența de lucru cea mai mică permisă de aplicație. Atunci când aplicația necesită o putere de calcul variabilă, este necesar să fie folosit un procesor dotat cu un sistem intern de management al puterii, v. secțiunea respectivă.

Două dintre cele mai simple porți CMOS, poarta inversoare și cea NAND sunt schițate în Fig. 1. Funcționarea este evidentă. Fig. 2 dă schema echivalentă de principiu a unui tranzistor MOS; capacitatea de grilă joacă un rol esențial atât în stabilirea frecvenței de lucru cât și în determinarea puterii consumate. O capacitate C_G mare necesită un timp mai mare de încărcare și înmagazinează mai multă energie electrostatică. Parte din energia necesară încărcării și parte

din energia înmagazinată de C_G se va pierde pe rezistențele din circuit la încărcarea / descărcarea C_G, de unde relația frecvenței cu puterea consumată.

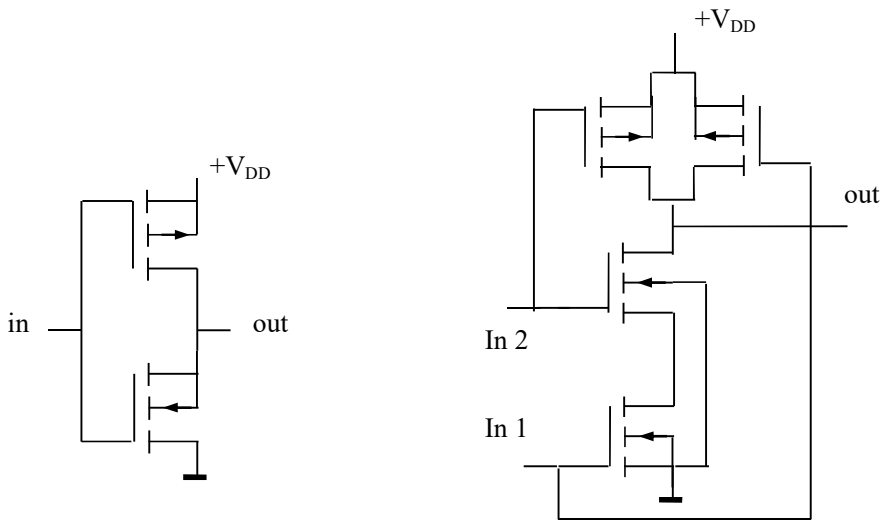


Fig. 1. Inversor CMOS Poarta NAND CMOS

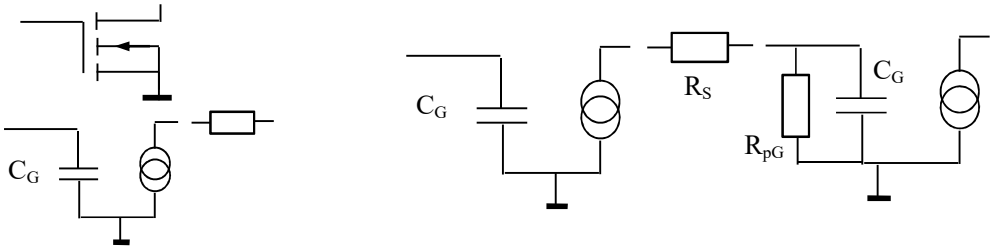


Fig. 2. Un tranzistor care comandă alt tranzistor – schema echivalentă simplificată, cu capacitatea de grilă încărcată de tranzistorul anterior prin rezistența serie R_S ce include rezistența de drenă și alte rezistențe parazite

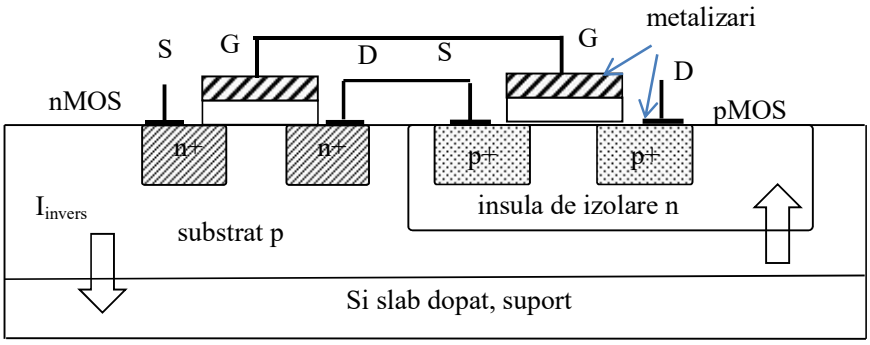


Fig. 3. Structură cu o poartă CMOS inversoare (2 tranzistori MOS-complementari), substrat de tip p și insulă de izolare tip n

Structura de principiu, în secțiune, a unei porți inversoare CMOS la nivel de plachetă de siliciu este ilustrată în Fig. 3. *G* semnifică grila și corespunde unei structuri formată dintr-un strat de izolator și o metalizare. Schița *nu* este la scară; de exemplu, grosimile straturilor structurii grilei sunt mult exagerate, iar substratul este mult mai subțire decât în realitate, față de zonele impurificate de tip n și p.

1.2. Consumul dinamic de putere a porții CMOS

Există două efecte principale care contribuie la pierderile dependente de frecvență într-un procesor. Aceste efecte vor fi discutate pe rând și rezultatele lor sumate în finalul discuției.

Efectul regimului activ (‚liniar') de funcționare a tranzistoarelor pe durata comutației

Deși ideal poarta CMOS corespunde la doi comutatori, dintre care unul este închis și celălalt deschis, astfel încât tot timpul ieșirea este binară, în realitate fiecare trecere dintr-o stare binară în alta are o durata semnificativă. Pe această durată, tranzistoarele se află în regim activ de funcționare și consumă un curent neneglijabil la o cădere de tensiune neneglijabilă pe tranzistor. Situația este indicată în fig. 4.

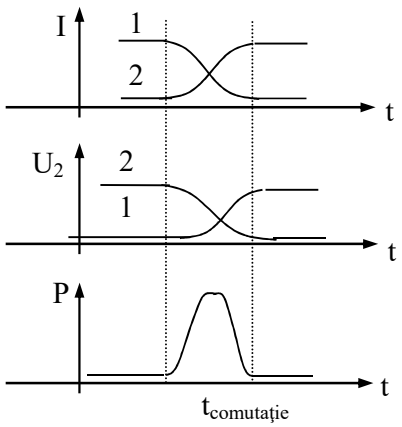


Fig. 4. Variația tensiunii și curentului pe cei doi tranzistori ai porții CMOS și puterea instantanee disipată pe poartă, având un maxim pe durata comutației

Puterea instantanee $P(t) = U(t)I(t)$ disipată pe durata comutației, respectiv puterea de comutație $P_{comut\_Q} = \frac{\int_{t_{comutatie}} U(t)I(t)dt}{t_{comutatie}}$ sunt mult mai mari decât puterea instantanee / puterea în starea ‚0' sau ‚1' a porții. Evident, această putere depinde de tensiunea de alimentare, $P_{comut} = P_{comut}(U)$. Puterea echivalentă consumată de circuit datorită exclusiv procesului de comutație este:

$$P_{dinamic_Q}(n_{porti}) = n_{porti} f\ P_{comut-Q}$$

unde n_{porti} este numărul de porți care comută pe secundă și f este frecvența de comutație. Graficul acestei componente a puterii (dinamice) consumate, funcție de frecvență este liniar, v. Fig. 5. Ulterior, va fi analizată și o altă componentă a puterii dinamice consumate, produsă de procesul de încărcare a condensatoarelor de grilă.

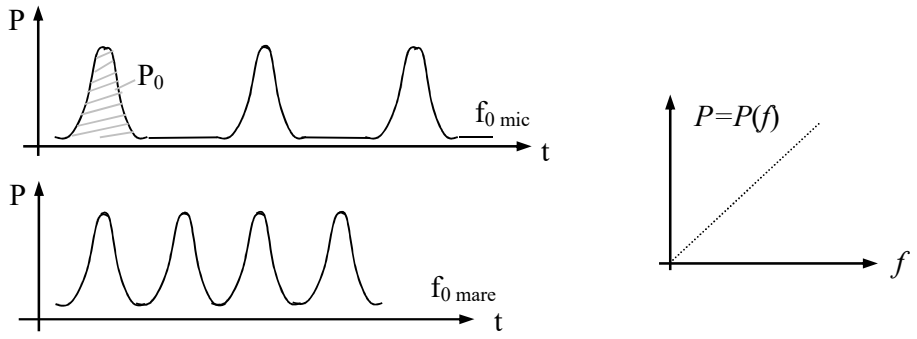


Fig. 5. Schița pentru calculul de putere consumată în regim dinamic (componenta puterii consumate datorită regimului activ al tranzistoarelor, $P_{dinamic-}$). Graficul pentru relația $P_{dinamic-Q} = fP_{comut}$.

Efectul dinamic al pierderilor la încărcarea și descărcarea capacităților de grilă

Încărcarea și descărcarea capacităților de grilă (condensatorilor formați între electrodul de siliciu și cel al metalizării grilei și având ca dielectric oxidul / izolatorul grilei) se face prin circuite cu pierderi, v. fig. 2 (ca principiu general). La fiecare ciclu de încărcare-descărcare, o parte $0 < 2\alpha < 1$ din energia înmagazinată de capacitatea grilei este pierdută. Cum energia pe un condensator depinde pătratic de tensiunea de încărcare, $W = Q^2/2C = CU^2/2$, energia pierdută per ciclu este $P_0 = \alpha CU^2$; în această relație putem folosi ca o

aproximare $U \approx U_{cc}$. La o frecvență de comutare f, această componentă dinamică, P_{din-C}, a pierderilor este

$$P_{din-C} \approx \alpha f C U_{cc}^2 .$$

Combinând cele două pierderi dinamice, obținem

$$P_{din-t} \quad = P_{comut\_Q} + P_{din-C} \approx f\, P_{comut-Q} + \alpha f C U_{cc}^2$$

Dacă ținem cont și de numărul de porți care comută, obținem

$$P_{din-tot}(n_{porti}) \approx n_{porti} f\left(P_{comut-Q}(U_{cc}) + \alpha C_G U_{cc}^2\right).$$

Din relația de mai sus, deducem că:

i) Puterea dinamică crește proporțional cu frecventa de tact;

ii) Puterea dinamică crește neliniar cu tensiunea de alimentare, printr-o componentă aproximativ liniară, $P_{comut-Q}(U_{cc})$ și printr-o componentă pătratică, $\alpha C U_{cc}^2$;

iii) Puterea dinamică crește aproximativ pătratic cu dimensiunea tranzistoarelor (proporțional cu aria lor), prin termenul $\alpha C_G U_{cc}^2$, deoarece $C_G \sim aria\ grilei$.

iv) Puterea dinamică depinde de numărul de porți efectiv comutate între stări; ca urmare, depinde de ce instrucțiuni se execută, de numărul de accesări ale memoriei de date etc. Altfel spus, puterea dinamică depinde de programul rulat și de modul de scriere a acestui program! Ca exemplu, este consumată mai puțină energie dacă pentru a produce o întârziere se execută operația *nop* repetat, decât dacă se repetă de același număr de ori o citire (fără alt efect) din memorie.

v) Fiecare instrucțiune are o ‚*semnătură de putere consumată*' specifică, dată de numărul de porți comutate pentru execuția ei.

Efectul regimului saturat-blocat de funcționare a tranzistoarelor

In calculul puterii pe poarta CMOS, trebuie adăugată și puterea consumată în stările saturat-blocat ale tranzistoarelor. Această putere este însă mică deoarece la saturație, în poarta CMOS, I_{sat} este mare, dar U_{sat} este mic, pe când în starea blocat, I_{bl} este neglijabil (pA…nA), iar U_{bl} este mare; ca urmare

$$P_{sat-bl} = I_{sat-Q}\ U_{sat-Q}\ + I_{bl-Q2} U_{bl-Q2} \approx 0.$$

Puterea în acest regim, pentru un întreg circuit, este de ordinul μW și $P_{sat-} \ll P_{din-tot}$.

1.3. Puterea statică disipată

Chiar și în absența tactului, circuitele CMOS au pierderi prin mai multe efecte:

Pierderi prin dielectricul grilei

Aceste pierderi depind de tensiunea aplicată și de temperatura ambiantă, conform cu:

$$I_{ox} = K\left(\frac{U_G}{d_{ox}}\right)^2 e^{-\gamma d_{ox}/U_G}.$$

unde U_G este tensiunea pe grila, d_{ox} este grosimea oxidului grilei, iar γ, K sunt constante. Relația de mai sus (conform literaturii) arată că pierderile prin grilă, $P_G = U_G I_{ox}$ cresc exponențial la scăderea grosimii grilei, ceea ce este o limitare în perfecționarea tehnologiei CMOS prin reducerea tensiunii de lucru (care necesită grile mai subțiri).

Pierderi prin curenți prin diodele polarizate invers

Aceste diode apar, conform figurii 3, între insulele de izolare si substrat, respectiv între substrat și suportul slab dopat. Deși densitățile acestor curenți sunt foarte mici, aria mare a acestor diode parazite face ca efectul curenților respectivi să nu fie neglijabili. Acești curenți sunt una din principalele cauze ale puterii absorbite în starea „*sleep*”. Efectul lor este cu atât mai mare cu cât crește temperatura: curenții inverși cresc exponențial cu temperatura plachetei.

1.4. Frecvența maximă de tact în relație cu temperatura ambiantă

Sunt două efecte care limitează, din punctul de vedere al disipării puterii, frecvența maximă de lucru la un procesor: creșterea puterii disipate la creșterea frecvenței de lucru și respectiv limitarea puterii transferate de la procesor către mediu de către rezistența termica R_{th-c-a} de la placheta de siliciu la mediu, în conjuncție cu valoarea temperaturii ambiante și cu temperatura maximă admisă a circuitului (v. Fig. 6). Primul efect este datorat pierderilor dinamice, prezentate în subsecțiunile anterioare.

$$P_{\max} = \frac{T_{\max} - T_{ambiant}}{R_{thj-amb}}$$

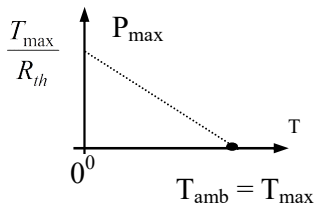


Fig. 6

Creșterea temperaturii plachetei duce la un efect nedorit: creșterea puterii statice prin creșterea exponențială cu temperatura a curenților prin diodele polarizate invers, precum și o creștere a puterii dinamice prin creșterea curenților de pierderi prin rezistențele echivalente ale condensatorilor de grilă, compensate de scăderea rezistențelor zonelor semiconductoare, precum rezistența drenei.
Neglijând efectele secundare precum variația rezistențelor parazite cu temperatura,

$$P_{static} + f_{max}\, UP_{comut} + \lambda f_{max} U^2 = \frac{T_{max} - T_{amb}}{R_{th-c-a}}$$

de unde,

$$f_{max} = \frac{\frac{T_{max} - T_{amb}}{R_{th-c-a}} - P_{static}(T_{max})}{UP_{comut} + \lambda U^2}$$

ceea ce arată că frecvența maximă este legată de tensiunea de alimentare și de puterea maximă disipată printr-o relație de forma

$$f_{max} \sim \frac{1}{UP_{comut} + \lambda U^2}$$

Frecvența maximă este limitată și de timpii de încărcare și descărcare a capacităților grilelor, anume

$$T_{min} = \frac{1}{f_{max}} > k(t_{inc} + t_{desc})$$

O alegere rezonabilă este $k > 2$. Din relația de mai sus, $f_{max} < \frac{1}{k(t_{inc}+t_{desc})}$ și folosind și relația anterioară,

$$f_{max} < \min\left(\frac{1}{k(t_{inc} + t_{desc})}, \frac{\Lambda}{UP_{comut} + \lambda U^2}\right)$$

unde Λ depinde de tehnologia folosită, capsulă și alți parametri de fabricație.

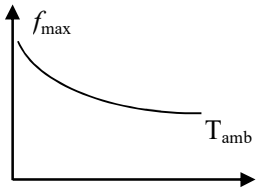


Fig. 7. Variație de principiu a frecvenței maxime de tact cu temperatura

Discuția de mai sus se poate rezuma schematic prin regulile (v. Fig. 7):

- Creșterea temperaturii ambiante limitează frecvența maximă de tact, care nu mai rămâne cea dată in catalog (la temperatura nominală).
- O răcire deficitară a circuitului conduce la reducerea frecvenței maxime de lucru (prin creșterea rezistenței termice echivalente).

1.5. Metode de optimizare a consumului în microsisteme – microsisteme avansate (de ex., ARM) - continuare

Ne reamintim că

$$P_{dinamic} = AU_a^2 f$$

unde f este frecvența de tact, U_a este tensiunea de alimentare, iar A o constantă. Constanta A depinde atât de circuit cât și de programul rulat (instrucțiuni). Puterea totală consumată este:

$$P_{total} = P_{dinamic} + P_{static}(in\ mare\ parte, prin\ diodele\ de\ substrat).$$

După modul în care realizează optimizarea consumului, microsistemele se pot clasifica în:

- Fixe, fără optimizare
- Adaptive și /sau reconfigurabile, cu management al puterii consumate:
    - Adaptive numai privind frecvența de tact
    - Adaptive privind atât frecvența de tact cât și tensiunea de alimentare (necesar la frecvențe mari)
    - Reconfigurabile (cu tact si/sau alimentarea selectiva a blocurilor necesare pentru operațiile curente). De exemplu „In the ARM1136JF-S processor extensive use is also made of gated clocks and gates to disable inputs to unused functional blocks.

Only the logic actively in use to perform a calculation consumes any dynamic power." [Catalog, ARM1136JF-S]. Explicații la curs.

- Combinații ale celor de mai sus.

La prima clasă (de ex., PIC16FXXX), optimizarea consumului depinde exclusiv de proiectant și se face prin modificarea tensiunii de alimentare în limitele permise, prin alegerea corectă a frecvenței de tact și prin optimizarea codului (v. partea B a cursului).

La sistemele adaptive, alegerea frecvenței de lucru se face automat, funcție de programul rulat. Deoarece frecvența maximă de lucru depinde de tensiunea de alimentare (crește cu tensiunea), majoritatea sistemelor adaptive folosesc un control simultan al frecvenței de lucru și al tensiunii de alimentare.

De ex., microsistemele ARM au incorporat un „manager de putere" care gestionează adaptările, funcție de cerințele de program.

Exerciții.

1. Estimați f_{tact} necesară la un controler 16FXXX care achiziționează și pune în memorie 200 de eșantioane (pentru cel puțin o perioadă de semnal ECG). Ce influenta are frecventa maxima din spectrul semnalului prelucrat asupra frecventei de tact a microsistemului ?

---

2. Optimizarea consumului prin optimizarea algoritmilor și programelor

Obținerea unui consum mic de putere se face printr-o combinație dintre consumul mic al circuitului (microsistemului) și un program (software) conceput cu obiectivul de minimizare a consumului. In acest capitol analizam – pe baza de exemple – modul în care un program poate fi îmbunătățit (optimizat) pentru un consum redus. Exemplele prezentate sunt variante de programe pentru rezolvarea aceleiași probleme practice. Sunt discutate în paralel performantele energetice ale algoritmului ales, ale codului ce implementează algoritmul, precum și performantele de timp și de memorie necesară ale lor.

In prima parte a capitolului introducem câteva concepte generale, apoi prezentăm problema de rezolvat și variantele de rezolvare.

Conceptul de complexitate a calculului (complexitate de timp și spațială/memorie)

Notație pentru complexitate: $O(n)$, unde n este dimensiunea intrării. Specifică modul de variație (funcțională) a necesarului de timp (a numărului de operații), respectiv a necesarului de memorie pentru implementarea unui algoritm, atunci când setul de date are dimensiune variabilă (crește spre infinit). Complexitate: logaritmică ($O(n) = log\, n$), liniară, pătratică ($O(n) = k \times n^2$), … polinomială, …, exponențială, …

Remarca 1. Numărul de operații, deci durata rulării unui program este proporționala cu complexitatea de timp.

Remarca 2. Puterea consumată este dependentă de complexitatea în timp (pentru o frecvență dată de tact).

Remarca 3. Constanta k ce multiplică funcția de complexitate și „supraîncărcarea" (constanta q, ce reprezintă, de obicei, operațiile de pregătire a lucrului) sunt necesare pentru a obține numărul de operații efectiv, $N(n) = kO(n) + q = k\, log\, n + q$, $N(n) = kO(n) + q = k \times n^2 + q$. Numărul $N(n)$ este important în determinarea celui mai bun algoritm sub raportul puterii consumate și a timpului de execuție, pentru seturi de date reduse ca dimensiuni.

Exemplu. Algoritmul de sortare prin căutare directă (comparare a valorilor două câte două) are complexitate pătratică, în timp ce algoritmul de sortare prin rearanjare („metoda bulei") are complexitate $O(n) = n \log n$.

Un exemplu

Să considerăm problema liniarizării unui senzor / traductor. Problema apare la realizarea de diverse instrumente de măsură.

----------=

Aici reamintit schema punte rezistivă + explicații / deducere formule (la curs)

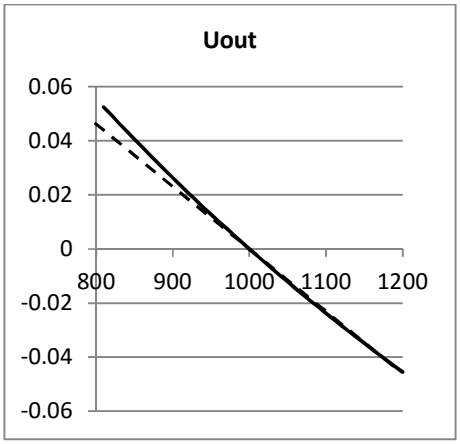
----------=

De exemplu, pentru o punte rezistivă, valoarea tensiunii de ieșire funcție de valoarea rezistenței senzorului este

$$U_{out} = U_0 \times \left(\frac{1}{2} - \frac{r_x}{(r_x + R_0)}\right).$$

Expresia $U_{out} = U_{out}(r_x)$ nu este liniară (ci rațională). Dorim ca ieșirea să fie proporțională cu valoarea rezistenței r_x, care poate fi, de exemplu, o

termorezistență. Aducerea la o lege proporțională se numește liniarizare. Dorim liniarizarea pe un interval de valori ale lui r_x, anume $r_x \in [r_1, r_2]$, cu o precizie dată (de problemă), de ϵ [V].



Există mai multe posibilități de liniarizare.

Varianta liniarizării directe

In cazul în care există o dreaptă $\delta: y = ar_x + b$ astfel încât, peste tot (pentru orice r_x) să fie satisfăcută condiția $|U_{out}(r_x) - y(r_x)| < \epsilon$, atunci problema liniarizării se rezolvă prin această metodă (aproximare). Avantajul este că metoda este relativ rapida (față de cazul unei aproximări polinomiale) și că este necesara putina memorie (doar pentru constantele a și b). Metoda este avantajoasă în special când constanta a poate fi reprezentată printr-o putere a lui 2, chiar dacă în acest scop se face un compromis între eroarea de aproximare și viteza de calcul / puterea consumată). Detalii la curs.

Exercițiu. Să se rezolve complet problema folosind criteriul erorii maxime locale,

$$ma_x |f(x) - ax - b| = \begin{cases} |f(x_1) - ax_1 - b| \\ |f(x_0) - ax_0 - b| \\ |f(x_2) - ax_2 - b| \end{cases}$$

unde x_0 este dat de condiția de extrem local în intervalul $[x_1, x_2]$,

$$\frac{d}{dx}|f(x) - ax - b| = 0\,.$$

Din condițiile de mai sus rezultă constantele dreptei de aproximare, a, b.

Obs.: In general, în manuale și programele de aproximare disponibile pe Internet se folosește alt criteriu de aproximare, cel al erorii pătratice globale minime; în acel caz, dreapta este determinată prin metoda cunoscută sub denumirea pe scurt „a celor mai mici pătrate”.

Varianta LUT- analiză

O variantă este folosirea unui LUT (look-up table), tabel de conversie, păstrat în memorie. De exemplu, făcând calculele, obținem tabelul de valori :

Metoda consumă memorie (cu atât mai multă cu cât gama $[r_1, r_2]$ este mai largă și cu cât precizia cerută (de problemă), ϵ, este mai mică.

Citirea tabelului de valori implică, de asemenea, timp mare și un număr relativ mare de operații. In cel mai simplist caz, după citirea valorii U_{out}, este necesar să se identifice intervalul în care valoarea se află, deoarece fiecare interval este asociat cu câte o valoare de ieșire, aplicând o căutare a intervalului. De exemplu, pentru eroarea ϵ, partiția intervalului $[r_1, r_2]$ se poate alege de forma (pentru simplitate, exemplificăm, în mod fals însă, pentru r, nu pentru U_{out} – tema de casă!):

$$[r_1, r_1 + \epsilon, r_1 + 2\epsilon, \dots, r_1 + n\epsilon, r_2]$$

unde $r_1 + (n+1)\epsilon \geq r_2$. Pentru o valoare determinată (de la CAD) a U_{out}, se caută pe rând, într-o buclă, intervalul în care se află valoarea U_{out}. Complexitatea este liniară, deoarece numărul de operații efectuate (comparări) este proporțional cu numărul de intervale (alternativ, cu inversul erorii admise).

In concluzie, căutarea sub-intervalului în care se află valoarea citită este consumatoare de timp și de energie. Ca urmare, deși pare simplă și este găsită în numeroase manuale, metoda poate fi foarte dezavantajoasă (consum de memorie, execuție lentă, energie consumată mare). Soluția este acceptabilă pentru instrumente de laborator, la care consumul nu este foarte important și pentru care procesorul are oricum o memorie suficient de mare.

Exercițiu. Puneți corect problema pentru (sub-)intervalele / partiția intervalului pentru U_{out}. Cum se determină valoarea erorii admise, ε, pentru tensiuni, dacă se știe eroarea admisă, ϵ, pentru rezistență ?

Implementarea efectivă a tabelului de conversie se poate face prin instrucțiuni, folosind memoria RAM, sau folosind memoria ROM. Varianta bazată

pe instrucțiuni care conțin constante are avantajul vitezei de lucru mai mari și consumului redus de memoria RAM (care adesea este mică la microcontrolerele de gamă joasă) – de exemplu, la controlerele PIC, regăsim valoarea dorită direct în acumulator. Cea mai directă cale de implementare este să folosim MOVLW k, unde constanta k este cea căutată în tabel. Un exemplu, bazat pe CALL (unde CALL cheamă rutina de conversie) și RELW k (unde k este constanta căutată) este dat în http://www.mikroe.com/chapters/view/10/chapter-9-instruction-set/.

Verificarea apartenenței valorii la un interval se poate rezolva mai simplu prin determinarea identității valorii citite cu una dintre valorile din intervalul prestabilit $[U_{out1}, U_{out2}]$, de exemplu, prin scădere (substracție) și verificarea bitului Z din registrul STATUS.

Detalii despre utilizarea (programarea) cu LUT pot fi găsite, de ex., în Abel Raynus, Tables ease microcontroller programming, EDN (Electronic Design News), April 22, 2010, p. 75, http://m.eet.com/media/1124407/6726481.pdf.

Varianta inversării funcției - analiză. O altă soluție este să inversăm funcția (să rezolvăm ecuația în r_x),

$$r_x = R_0 \frac{g(U_{out})}{1 - g(U_{out})}, \qquad g(U_{out}) = 0.5 - \frac{U_{out}}{U_0}.$$

Inversarea funcției presupune însă multiplicări și divizări:

- $g(U_{out}) = 0.5 - \frac{U_{out}}{U_0}$ - o multiplicare, cu constanta $\frac{1}{U_0}$,
- $\frac{g(U_{out})}{1-g(U_{out})}$ o divizare,
- O multiplicare cu R_0.

Unul dintre dezavantajele metodei este că necesită un procesor (CPU) pe un număr mare de biți, pentru ca erorile de calcul produse prin trunchieri la multiplicare și divizare și acumulate continuu pe parcursul calculului să fie în limitele acceptabile. Pentru procesoarele care nu au capabilități de virgulă mobilă este necesar să se lucreze cu valori multiplicate, de ex.,

$$G = 16g(U_{out}) = 8 - \frac{16U_{out}}{U_0}$$

$$g(U_{out}) = G/16$$

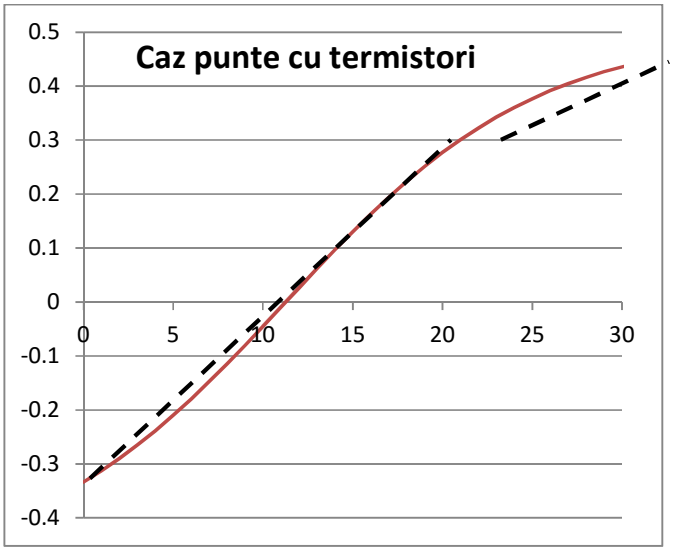
$$R_x = 16r_x = R_0 \frac{16g(U_{out})}{1 - g(U_{out})},$$

$$r_x = R_x/16\,.$$

Factorul de scalare (mai sus, egal cu 16) se alege funcție de problemă. Creșterea precizie prin scalari cu puteri ale lui 2 duce însă la creșterea și a timpului de calcul și a energiei consumate.

O metodă analitică prin segmente de dreaptă

In unele cazuri, variația valorii măsurate efectiv de DAC, în raport cu cea de intrare, nu se poate aproxima satisfăcător cu un segment de dreaptă; un asemenea caz este cel al unei punți cu termistor, vezi Fig. 2.



Amprenta de timp și energetică a unor operații algoritmice

Pentru a clarifica discuția, care se referă exclusiv la procesoare RISC, vom numi operație-μ o operație din setul microsistemului folosit și prin operație-A o operație folosită în algoritmi și limbaje de programare de nivel înalt. De exemplu, înmulțirea este o operație-A, dar nu și o operație-μ pentru un microcontroler din seria 16FXXX. Știm că unele operații, anume cele de salt, necesită timpi diferiți la un procesor RISC. Operațiile de salt necesită și consum diferit (mai mare) de energie decât operațiile standard, căci se „consumă” doi cicli mașina pentru realizarea lor, cu consum suplimentar energetic pe cei doi cicli, datorită operației de golire (realiniere) pe *pipe-line*. Ca urmare, la sistemele cu consum redus, operațiile de salt se evită dacă este posibil.

Întreruperile sunt consumatoare și de timp (prin necesitatea rutinelor de deservire, ISR, – intrare și ieșire din întrerupere) și de energie (dependent de numărul de operații / cicli mașina efectuați în ISR).

3. Exemple de exercitii (numerotare imperfecta in aceasta versiune)

1. Explicați în câteva rânduri metoda și dați un scurt exemplu de construire a codurilor binare pentru operațiile unui MC RISC, așa cum s-a făcut pentru MC EASI 201X în clasă. Considerați că MC are 27 de operații, care includ operații din toate clasele de la 16FXXX, iar memoria de date are un singur banc, de 516 cuvinte. Comparați soluția dată cu cea folosita la PIC 16F84.
2. Cum se modifică puterea consumată în funcție de tensiunea de prag a tranzistoarelor CMOS? Pentru reducerea puterii consumate este bine ca U_{prag} să fie mică sau mare? Explicați.
3. Toate instrucțiunile absorb aceeași putere / același curent mediu de la sursă? Pe parcursul executării unei instrucțiuni curentul consumat de la sursă este constant? Explicați.
4. Procesorul ARM Cortex M0 are un circuit (subsistem) de management al puterii. Explicați pe scurt ce funcții are un asemenea circuit, modurile în care el reduce consumul de putere și scrieți relațiile principale pe care funcționarea se bazează (care justifică un asemenea subsistem).
5. Ce este latența întreruperii? Care sunt factorii care determină latența? Depinde latența efectivă a întreruperii (timpul între momentul apariției cererii și momentul în care prima linie de cod ce deservește efectiv cererea) de programul rulat? Poate fi aceasta redusă prin program? Cum – dați un exemplu.
6. Consideram un microcontroler pentru care există 3 instrucțiuni de chemare de subrutine, CALL, GO TO și o instrucțiune de chemare condiționata CALLIF de forma “daca bit ‘b’ din W1 = 1, CALL Instr. Cați biți ar necesita cuvântul MP la un microcontroler de tip PIC , pentru a realiza aceasta instrucțiune? W1 este un cuvânt din MD.

Analizam opcodul necesar:

- 2 biți pentru clasa de instrucțiuni de tip CALL (la PIC)

- 2 biți pentru identificator în cadrul clasei (sunt 3 instr în clasa)
- 3 biți pentru adresa ‘b’
- 7 biți pentru adresare W1

Total parțial 14 b

- 11 biți pentru adresa de salt

Total 25 biți

7. Considerăm o extensie a microcontrolerului 16F84 în care am avea instrucțiuni logice și aritmetice cu încărcare de doi operanzi din MD și transfer a rezultatului la o adresa oarecare în MD. Cați biți ar trebui să aibă cuvântul din MP?

- 2 biți clasa operații
- 4 biți identificare în cadrul clasei
- 7 biți adresa operand A în MD
- 7 biți adresa operand B în MD
- 7 biți adresa rezultat în MD

TOTAL 27 biți !

8. De ce timpul de acces al memoriilor este mare? Justificați

- circuit combinațional pe multe niveluri (log n, n= nr de cuvinte)
- necesar registru menținere cuvânt adresare
- necesar registru menținere cuvânt citit (instrucțiune)
- necesar selectare mod lucru (scriere/citire la RAM)
- timp efectiv de citire cuvânt

9. Creste sau scade frecvența de tact / lucru a unui microcontroler la creșterea capacității MP ? Justificați.

R: Creste timpul circuitului combinațional de selectare adresa

10. Ce tip de tranzistori se folosesc tipic în u-sisteme? De ce?

11. Precizați care a fost primul microcontroler și când a fost fabricat

12. Precizați în detaliu ce este stiva, funcția și arhitectura (structura) ei, precum și cu ce alte blocuri este conectata.

Precizați în detaliu formatele a trei clase de instrucțiuni tipice.

13. Ce fel de porți sunt utilizate în u-sisteme? Explicați efectul întârzierilor pe porți asupra funcționarii u-sistemelor.

14. Explicați în câteva rânduri funcționarea de tip pipeline

15. Considerați o instrucțiune numita AND_A_M care face SI logic între un cuvânt din acumulator și un cuvânt din memorie și păstrează rezultatul în acumulator. Considerați un ALU de 8 biți. A) Ce alte info aveți nevoie pentru a spune câți biți sunt necesari pentru opcodul operației? Precizați acele info și propuneti un opcod, justificat.

16. Desenați structura unui uP a cărui ALU are 16 biți și care are o memorie de program de 4 kbytes. Desenați numai blkocurile ALU- MP –stiva – PC – magistrale corespunzătoare, pe baza modelului simplu de la 16F84.

17. Precizați semnificația acronimelor (prescurtărilor) CISC și RISC.

18. Explicați ce legătură este între dimensiunea (numărul de biți) ai numărătorului de program și numărul de biți ai unei locații (cuvânt) din memoria de date și respectiv numărul de biți ai memoriei de program?

19. Dorim să dotăm procesorul EASI2013 de tip RISC cu instrucțiuni aritmetice și logice la nivel de cuvânt, operațiile fiind făcute cu valori din memorie și cu constante, instrucțiuni de deplasare, instrucțiuni la nivel de bit, precum și instrucțiuni de transfer între acumulator și memorie. A) Propuneți un asemenea set de operații (folosind exemplul 16FXXX) și apoi precizați pe scurt cum ați defini ‚optim' codarea informațiilor. B) Dacă procesorul este de 8 biți, memoria de date este organizată pe 4 file (bank-uri) de 128 de cuvinte, iar adresarea se face (prin instrucțiuni) numai la nivel de filă (bank), ce dimensiune (număr de biți) are cuvântul din memoria de program?

20. Se cunoaște durata de răspuns la un bistabil rapid de 0.5 ns, iar la o poartă logică de 0.1 ns. Cu acestea se face o memorie pentru care circuitul de decodificare de adrese are 3 niveluri. Care este cel mai rapid tact utilizabil cu această memorie?

R:

Circ de decodificare de adrese necesită 3 x 0.1 ns = 0.3 ns.

Circ de tip registru de adrese necesită 0.5 ns.

Circ de memorie propriu-zisă necesită 0.5 ns.

Circ de tip registru de ieșire necesită 0.5 ns.

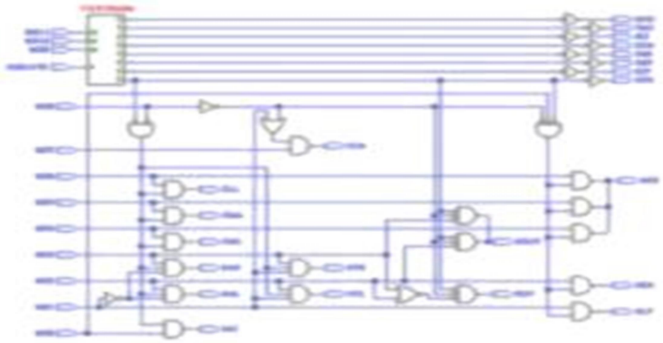
Total = 0.5 x 4 = 2 ns

Marja 0.5 ns

Total general 2.5 ns

Tact 2 x 2.5 ns = 5 ns

21. Precizați ce este un DSP și prin ce diferă de un procesor uzual. Scrieți ecuația unui filtru nerecursiv și explicați în conexiune cu aceasta ecuație.
22. Ce legătură este între numărul de biți ai unei locații (cuvânt) din memoria de date și respectiv numărul de biți ai memoriei de program? Explicați.
23. Să se precizeze numărul de locații (cuvinte) ale stivei necesare la un procesor care trebuie să admită programe cu până la 5 chemări de subrutine imbricate (una în alta).
24. Desenați un circuit de acceptare/blocare a întreruperilor pentru un procesor cu 3 surse de întrerupere (notate, de ex., A, B, C), neprioritizate. Oricare dintre întreruperi trebuie să poată fi acceptată sau blocată, iar circuitul trebuie să permită și blocarea tuturor întreruperilor cu un singur bit.
25. Ce rol are registrul de control al întreruperilor (notat INTCON la 16F8XXX)? Ce conține el?
26. O memorie de date (MD) are cuvinte de 16 biți și capacitate de 14 k cuvinte. Memoria e structurată pe file (bank-uri) de 4 k cuvinte. Procesorul EASI2013 dotat cu această memorie folosește instrucțiunea „Deplasează Acumulatorul la Locația N din MD”, mnemonic DAL *f*. Accesul la file se face cu 2 biți proveniți dintr-un registru de stare și cu ceilalți din instrucțiune, biții din instrucțiune fiind desemnați prin *ff…f*. Câți biți „*f*” sunt necesari în instrucțiunea DAL?
27. La Procesorul EASI2013, decodorul de instrucțiuni are o schemă de tipul din figură. Presupunând că acel circuit este chiar decodorul folosit și că fiecare poartă logică are un timp de propagare de 1 ns, să se determine dacă decodarea instrucțiunilor durează mai mult decât citirea memoriei de program, care se știe că durează 5 ns. (Se neglijează circuitul din stânga-sus).



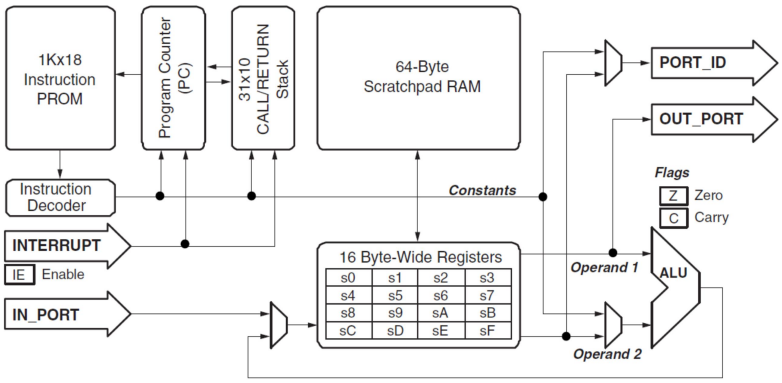
[din literatura]

21. Microcontrolerul EASI2013 are un registru de control al întreruperilor (notat **IntContr**) ca în fig.

Bit fanion al într. #1	Bit de accept al într. #1	Bit fanion al într. #2	Bit de accept al într. #2	Bit fanion al într. #3	Bit de accept al într. #3	Bit de accept general de într. (GIE)	Bit fanion de revenire din hibernare

Desenați un circuit logic care să corespundă acestui **IntContr**.

22. Microcontrolerul PicoBlaze™ din familiile de FPGA Spartan®-3, Virtex®-II, and Virtex-II Pro (Xilinx) (controlerul este înglobat în aceste FPGA) este un procesor 8-bit RISC și are un set de cca. 55 instrucțiuni. El include un set de 16 regiștri generali (regiștri de date) cu cuvinte de 1 byte și o memorie de program de 1K. Schema bloc este reprodusă după catalog în figură. ALU nu este dotat cu un registru acumulator special, ci folosește pentru păstrarea rezultatului unul dintre cei 16 regiștri, care pot fi regiștri de intrare și/sau acumulator, în instrucțiuni de tipul **ADD sX, sY**. O memorie de date de doar 64 de cuvinte este accesibilă direct pentru scrierea și citirea celor 16 regiștri, printr-o singură instrucțiune de tipul **FETCH sX, (sY)**, care citește locația din memorie de la adresa din registrul sY (din memorie) și o pune în registrul sX. Instrucțiunile (op-codurile) au 18 biți. A) Explicați de ce această structură este utilă ca DSP. B) Explicați cum s-ar putea implementa un filtru nerecursiv de ordin cât mai mare pe această structură. C) Explicați de ce se crește viteza de lucru (comparativ de ex. cu 16F84) la aceasta structură, în care acumulatorul pare a lipsi.



PicoBlaze™, © Xilinx

1. Descrieți modul de concepere a opcodurilor unui microcontroler și construiți un set de instrucțiuni cu opcodurile respective.
2. Tipuri de oscilatori de tact la 16FXXX.
3. Setul de instrucțiuni la 16FXXX – clase de instrucțiuni, explicații
4. Circuitul logic de întreruperi
5. - A) a) Se citește portul B și rezultatul se pune in bancul al doilea de memorie; b) se deplasează la stânga o data; c) se aduna valoarea citita cu un număr de la o locație din bancul intai de memorie si, daca ultimul bit este 0 se trimite la portul B (ieșire); c) se calculează durata de execuție, cunoscând f_{tact} = 1 MHz.[Se admit 2 erori de cod]

   - B) (a) Se generează pe portul B un semnal PWM cu perioada 80 us, cu factor de umplere astfel: 4 perioade cu $\eta = 50\%$, 8 perioade cu $\eta = 12{,}5\%$. (total 12 perioade). (b) Se monitorizează o cerere de întrerupere externa și când apare se reia generarea semnalului PWM.

1. Pipeline – descriere, avantaje; durata instrucțiunilor de salt
2. Descrieți relațiile dintre frecvența de lucru și puterea consumata la un uC. Tipuri de oscilatori de tact la 16FXXX.
3. Analiza tipurilor de întreruperi la 16FXXX
4. Descrieți structura generală a unei rutine de deservire a întreruperilor
5. Circuitul logic pentru întreruperi
6. - A) a) Se citește portul B și rezultatul se pune in bancul al doilea de memorie; b) se aduna valoarea citită cu un număr de la o locație din bancul intai de memorie; c) se deplasează la dreapta de 4 ori rezultatul si, daca

ultimul bit este 0 se trimite la portul B (ieșire); c) se calculează durata de execuție, cunoscând f_{tact} = 1 MHz. [Se admit 2 erori de cod]

- B) (a) Se generează pe portul B un semnal PWM cu perioada 40 μs, cu factor de umplere astfel: 4 perioade cu $\eta = 25\%$, 8 perioade cu $\eta = 50\%$. (total 12 perioade). (b) Se monitorizează o cerere de întrerupere externa și când apare se reia generarea semnalului PWM.

1. a) Încărcaţi în registrul Rx la locația 0x20 din bancul 1 din memoria de date constanta 1011 0010. b) Adunaţi valoarea din acea locaţie cu 0110 1111 şi rezultatul se pune în registrul Ry la locaţia 0x21 (din acelaşi banc). c) Verificaţi manual ce biţi se modifică în STATUS d) Dacă rezultatul ultimei operaţii este par, schimbaţi bancul de memorie şi scrieţi în locaţia pereche (0x21 în registrul Rz) din bancul 2 valoarea lui Ry/2, altfel se păstrează valoarea lui Ry decrementată.

2. să se genereze pe pinul RB2 al PORTB un semnal periodic cu impulsuri de durata 1 ms. (Tactul este de 4 MHz). Durata între două impulsuri consecutive este de 10 ms.
b) Modificaţi programul astfel încât simultan pe pinul RB3 să fie generat un semnal de aceeaşi perioadă, dar cu impulsuri de durată 2 ms.
La apăsarea unui buton se activează / dezactivează generarea semnalului / semnalelor. Să se folosească cel puţin o sursă de întreruperi.

e) Citiţi ultimii 4 biţi din STATUS şi transferaţi-i pe portul B0-B3 ca ieşire. f) Presupuneţi că pinii RB4-RB7 sunt conectaţi hard ca intrări astfel: B4 cu B0, B5 cu B1, B6 cu B2, B7 cu B3. g) Încărcaţi numărătorul cu conţinutul (B2,B1,B0) pe poziţia biților cei mai semnificativi, restul biţilor fiind zero. h) la terminarea numărării, scrieţi ‚0' pe ultimul bit (bit 0) din STATUS.
2. Să se realizeze un semnal PWM cu factor de umplere de 30%. Pentru generarea semnalului se va folosi pinul 3 al PORTB. Semnalul este periodic de perioadă T= 10 ms (se va explica cum s-a calculat timpul).

La apăsarea unui buton plasat pe pinul RB0 (asociat întreruperii externe) se va schimba factorul de umplere în 70%, respectiv la a următoarea activare a acestei rutine înapoi în 30%.

1. Un microcontroler (MC) are o întrerupere prioritară și mai multe întreruperi de același nivel (neprioritare). MC rulează un program principal în care sunt 2 salturi imbricate și care acceptă întreruperea prioritară și o (singură) întrerupere neprioritară. Rutina întreruperii prioritare include un salt (de ex. prin chemare de subrutina), iar rutina întreruperii neprioritare nu include salturi. A) să se schițeze graful executării programului (calitativ). B) să se precizeze numărul de locații pe stiva necesar și să se justifice.

7. A) Explicați pe scurt conceptul de „pipeline" (numit posibil la curs și „banda de asamblare"). B) Presupunem că la MC EASI201X citirea memoriei de program durează 30 ns, operațiile de decodificare durează 10 ns, execuția în ALU durează 25 ns, iar citirea / scrierea memoriei de date durează 70 ns. Considerați o instrucțiune de tip MOV W F („mută conținutul acumulatorului în memorie la locația F"). Propuneți o procedură de *pipeline* și justificați numărul de niveluri ales. C) Care va fi frecvența maximă de tact a procesorului EAI 2014 dacă presupuneți că pentru fiecare fază a procedurii *pipeline* este nevoie de 4 tacturi?
8. Procesorul ARM Cortex M0 are un circuit (subsistem) de management al puterii. Explicați pe scurt ce funcții are un asemenea circuit, modurile în care el reduce consumul de putere și scrieți relațiile principale pe care funcționarea se bazează (care justifică un asemenea subsistem).

V1 Scrieți un program care să genereze pe pinul B1 al 16F84 un semnal PWM cu factor de umplere 1/64. Alegeți f_{tact} astfel încât perioada să fie de 1 ms. Se comandă un motor de c.c. (sau un releu) cu un semnal PWM de pe un pin al MC 16F84. A) Curentul efectiv prin pin (motor, releu) depinde de factorul de umplere? Cum? (scrieți relația de bază). B) Curentul efectiv prin pin (motor, releu) depinde de frecvența semnalului PWM? De ce? Explicați calitativ cum. (Indicație: puteți face referire la spectrul semnalului ?)

La problema:

Se comandă un motor de c.c. (sau un releu) cu un semnal PWM de pe un pin al MC 16F84. A) Curentul efectiv prin pin (motor, releu) depinde de factorul de umplere? Cum? (scrieți relația de bază). B) Curentul efectiv prin pin (motor, releu) depinde de frecvența semnalului PWM? De ce? Explicați calitativ cum. (Indicație: puteți face referire la spectrul semnalului ?)

a) Banal, $I_{ef} = \frac{1}{T}\int_0^T i(t)dt = \psi U_a/R_s$ unde ψ e factorul de umplere, U_a tensiunea de alimentare, R_s rezistența echivalentă a sarcinii (înfășurare și pierderi in miez).
b) Pierderile in miez și cele in înfășurare depind de frecvență – cresc cu frecvența. Daca frecvența semnalului PWM crește, $R_{s-echivalent} = R_s(f)$ creste. Aceasta arată că relația de mai sus (pct. a) este valabilă doar pentru o frecvență dată și că este doar aproximativă, căci valoarea lui ψ modifică spectrul semnalului și deci și pierderile.

1. Pipeline – descriere, avantaje; durata instrucțiunilor de salt
2. Descrieți relațiile dintre frecvența de lucru si puterea consumata la un uC.
3. Tipuri de oscilatori de tact la 16FXXX.
4. Analiza tipurilor de întreruperi la 16FXXX
5. Un microcontroler (MC) are o întrerupere prioritară și mai multe întreruperi de același nivel (neprioritare). MC rulează un program principal în care sunt 2 salturi imbricate și care acceptă întreruperea prioritară și o (singură) întrerupere neprioritară. Rutina întreruperii prioritare include un salt (de ex. prin chemare de subrutina), iar rutina întreruperii neprioritare nu include salturi. A) să se schițeze graful executării programului (calitativ). B) să se precizeze numărul de locații pe stiva necesar și să se justifice.
6. Descrieți ce este o întrerupere și conceptele anexe (cerere de întrerupere, rutina de deservire de întrerupere). Explicați diferența față de alte evenimente și/sau secvențe de instrucțiuni ale unui program.
7. Presupunem că un MC are un singur numărător disponibil pentru utilizator (Timer0). Presupunem că programul principal al MC permite apariția unei singure întreruperi. De asemenea, presupunem că în programul principal se

execută, printre altele, adunări succesive, scăderi, precum și o buclă al cărei contor este menținut de Timer0. A) Determinați (pe baza precizărilor de mai sus) și explicați pe scurt ce regiștri (informații) trebuie să salvați la intrarea în întrerupere. B) Scrieți liniile de cod corespunzătoare salvării la intrarea în întrerupere și recuperării informației la ieșirea din întrerupere, pentru PIC 16FXXX.

8. Presupunem că programul principal al MC permite apariția unei singure întreruperi. De asemenea, presupunem că în programul principal se execută, printre altele, adunări succesive, operații logice (AND), precum și o buclă al cărei contor este menținut de Timer1. A) Determinați (pe baza precizărilor de mai sus) și explicați pe scurt ce regiștri (informații) trebuie să salvați la intrarea în întrerupere. B) Scrieți liniile de cod corespunzătoare salvării la intrarea în întrerupere și recuperării informației la ieșirea din întrerupere, pentru PIC 16FXXX. Determinați aproximativ (in cicli mașină) timpul dintre momentul sosirii cererii de întrerupere și momentul efectuării rutinei (operațiilor) propriu-zise ce deservesc efectiv întreruperea.
9. Explicați cum poate influența programul rulat puterea consumată de un MC.
10. A) Explicați rolul generatorului de tact (oscilatorului) intern. Ce frecvență tipică are? B) Explicați modurile tipice de protecție împotriva funcționării deficitare la un MC.
11. Scrieți relațiile principale care definesc puterea consumată de un MC și explicați ce sunt termenii puterii consumate, cui se datorează, precum și forma ecuațiilor, cu scurte explicații.
12. Explicați în câteva rânduri de ce și cum separarea pe *chip* (placheta de Si) a unor părți ale circuitului MC în „insule” cu alimentare separată și controlată separat poate permite reducerea puterii medii consumate.
13. Utilizarea *pipeline*-ului pe mai multe niveluri scade sau crește curentul absorbit de un MC? Crește sau scade curentul la creșterea numărului de niveluri de *pipeline*? Explicați.
14. Este vreo legătură între capsula folosită și frecvența maximă de lucru a unui procesor? Explicați.
15. Este vreo legătură între temperatura ambiantă și frecvența maximă de tact a unui MC? Ar trebui să își modifice MC frecvența (maximă) de tact în funcție de temperatura ambiantă? Explicați.

16. Un procesor are, uzual, un numărător atașat generatorului de tact intern. Dacă nu este inclus un divizor de tact (*pre-scaler*), câți biți trebuie să aibă numărătorul respectiv pentru ca să poată face resetarea sistemului după 0,1 secunde? Se consideră că generatorul intern are frecvența uzuală.
17. Trebuie să avem grijă la scrierea unui program ce acceptă întreruperi ca, la apariția unei întreruperi, să se salveze în memoria de date și conținutul numărătorului de program ? Explicați pe scurt răspunsul.
18. Cum poate să aibă acces (hard și *soft*) un traductor / senzor la un CPU dintr-un microcontroler ? Dați 1-2 exemple de senzor, acces hard (conectare) a senzorului la microcontroler și acces *soft* – mod de transmitere a datelor de la senzor la CPU. (Există mai multe răspunsuri corecte! Oricare este suficient.)
19. Se pot transfera valorile (cuvintele) de pe stivă în acumulator la MC 16FXXX ? Dar în memoria de date? Explicați răspunsul.
20. Se pot „citi" instrucțiunile executate de un microcontroler doar prin urmărirea curentului absorbit? Scurtă explicație.
21. Ce este o memorie *cache* și ce rol are?
22. La procesoarele ARM Cortex (și altele ARM) există un modul de gestionare a întreruperilor. Ce rol are? (Elemente generale).
23. Un numărător numără apariția de evenimente externe (de ex., trecerea unor piese pe banda de ambalare). Când ajunge la 32, cutia se umple și trebuie să se genereze o întrerupere a MC care controlează sistemul. Intre două asemenea cereri de întrerupere, MC execută un program principal care realizează sume de numere cu valori ce trebuie puse în memoria de date. Scrieți liniile de program care permit intrarea in întrerupere și salvarea acelor date care, din textul problemei, rezultă că sunt necesare la revenirea din întrerupere.
24. Se comandă un motor de c.c. (sau un releu) cu un semnal PWM de pe un pin al MC 16F84. A) Curentul efectiv prin pin (motor, releu) depinde de factorul de umplere? Cum? (scrieți relația de bază). B) Curentul efectiv prin pin (motor, releu) depinde de frecvența semnalului PWM? De ce? Explicați calitativ cum. (Indicație: puteți face referire la spectrul semnalului ?)
25. Scrieți un program care să genereze pe pinul B1 al 16F84 un semnal PWM cu factor de umplere 1/64.

26. Considerăm ca un consumator (sarcina) care necesita un curent de 1,5 ori mai mare decât curentul maxim debitat pe un pin de către MC trebuie alimentat cu un semnal PWM cu factor de umplere 1/64. Scrieți un program care să genereze acel semnal pe portul B, cu o scurtă explicație a alegerii pinilor folosiți.

2.a) Să se genereze pe pinul RB3 al portului B un semnal PWM cu factor de umplere de 30%, a cărui perioadă este T= 10ms. Pentru o frecvenţă de oscilator la alegere, justificaţi că duratele 0L, 1L sunt cele corecte.
b) Cum s-ar genera simultan pe pinul RB2 un semnal PWM cu factor de umplere 70%, (eventual la apăsarea unui buton aflat pe pinul RB0). Abordarea problemei citirii butonului se poate face utilizând întreruperi.

RASPUNSURI

La problema:
Se comandă un motor de c.c. (sau un releu) cu un semnal PWM de pe un pin al MC 16F84. A) Curentul efectiv prin pin (motor, releu) depinde de factorul de umplere? Cum? (scrieți relația de bază). B) Curentul efectiv prin pin (motor, releu) depinde de frecvența semnalului PWM? De ce? Explicați calitativ cum. (Indicație: puteți face referire la spectrul semnalului ?)

a) Banal, $I_{ef} = \frac{1}{T}\int_0^T i(t)dt = \psi U_a/R_s$ unde ψ e factorul de umplere, U_a tensiunea de alimentare, R_s rezistența echivalentă a sarcinii (înfășurare și pierderi in miez).
b) Pierderile in miez și cele in înfășurare depind de frecvență – cresc cu frecvența. Daca frecvența semnalului PWM crește, $R_{s-echivalen} = R_s(f)$ creste. Aceasta arată că relația de mai sus (pct. a) este valabilă doar pentru o frecvență dată și că este doar aproximativă, căci valoarea lui ψ modifică spectrul semnalului și deci și pierderile.

a) Încărcați la locația xxx din bancul 1 din memoria de date constanta 1011 0010. b) Adunați valoarea din acea locație cu 0110 1111 și rezultatul se pune in locația yyy. c) Verificați manual ce biți se modifica in STATUS. D) Citiți ultimii 4 biți din STATUS și verificați dacă suma ultimilor doi este pară, iar dacă da, schimbați

bancul de memorie și scrieți rezultatul de la punctul b) in locația pereche din celalalt banc.
e) Citiți ultimii 4 biți din STATUS și transferați-i pe portul B2-B5 ca iesire. f) Presupuneti ca portul B2-B5 este conectat hard astfel: B2 cu B1, B3 cu B6, B4 cu B7, ultimii conectați ca intrări (prin program). g) Încărcați numărătorul cu conținutul (B1,B6,B7) pe poziția biților cei mai semnificativi, restul biților fiind zero. H) la terminarea numărării, scrieti ‚0' pe ultimul bit (bit 0) din STATUS.

1. Sa se scrie codul in limbaj de asamblare și in C pentru a realize următoarele operații:

a) Se operează SAU intre continutul unei locații din bank-ul al doilea al memoriei cu constanta 11001100, se divizează prin 2 și se pune rezultatul in aceeasi locatie. Se repeata operatia in bucla infinita.
b) Daca pe pinul B3 se modifica valoarea in timpul executarii operațiilor de mai sus, atunci se operează ultima valoare obtinuta (aflata deja in memorie) prin SAU cu 00110011 și apoi se continua cu operațiile de mai sus (punctual (a)). Se va rezolva acest punct in doua variante: prin monitorizarea in bucla a pinului B3 și prin întreruperi.

c) să se determine timpul de răspuns la modificarea valorii pe pinul B3 in cele doua variante (monitorizare, întreruperi). fclock = 1 MHz (16F84).

2. Să se scrie codul în limbaj de asamblare și în C pentru a realiza următoarele operații:

a) Se preia de la portul B4-B7 un număr binar de 4 biți.
b) Se înmulțește de 4 ori succesiv cu 2.
c) Se pune în memorie rezultatul la o locație în fila a doua (bank-ul al doilea).
d) Se adună cu constanta zecimală 5 și se determină dacă apare depășire. La depășire se pune pe 1 pinul B3.
e) Se reiau de 4 ori operațiile începând cu (a).
f) Estimați durata executării programului, dacă $fclock$ = 1 MHz (16F84)

1. Să se scrie codul în limbaj de asamblare și în C pentru a realiza următoarele operații:

a) Se preia de la portul B0-B7 un număr binar.

b) Se determină dacă numărul este par sau impar.

c) Dacă numărul este par, se aplica operația SAU între număr și constanta zecimală 9, iar la rezultat se adună constanta zecimală 5.

d) Rezultatul se pune în memorie la o locație în fila a doua (bank-ul al doilea).

e) Dacă numărul citit este impar, se împarte la 2, iar rezultatul se pune în memorie.

f) Estimați durata executării programului, dacă $fclock$ = 1 MHz (16F84), pentru cazul numărului impar la punctul (b).

2. Să se scrie codul în limbaj de asamblare și în C pentru a realiza următoarele operații:

a) Se operează SAU între conținutul unei locații din fila a doua a (bank-ul al doilea al) memoriei cu constanta 11001100, se divizează prin 2 și se pune rezultatul în aceeași locație. Se repetă operația în buclă infinită.

b) Dacă pe pinul B3 se modifică valoarea în timpul executării operațiilor de mai sus, atunci se operează ultima valoare obținută (aflată deja în memorie) prin SAU cu 00110011 și apoi se continuă cu operațiile de mai sus (punctual (a)). Se va rezolva acest punct în două variante: prin monitorizarea în buclă a pinului B3 și prin întreruperi.

c) Să se determine timpul de răspuns la modificarea valorii pe pinul B3 în cele două variante (monitorizare, întreruperi). Se dă $fclock$ = 1 MHz (16F84).

3. Să se scrie codul în limbaj de asamblare și în C pentru a realiza următoarele operații:

a) Se generează continuu pe pinul B2 impulsuri de durata a 50 de cicli mașină și factor de umplere 1/25.

b) La pinul B3 se conectează punctul comun al unui rezistor și al unui comutator; rezistorul este conectat la +5V cu un capăt, iar comutatorul (un buton) este conectat la masă. La închiderea comutatorului, a cărui poziție normală este „circuit deschis”, microcontrolerul trebuie să genereze 5 impulsuri cu factor de umplere variabil, în ordinea 1/25, 2/25, 3/25, 4/25, 1/5. Rezolvați prin monitorizarea în buclă a pinului B3.

c) Circuitul comutatorului nu este bine ales. Ce se va putea întâmpla „în lumea reală” la apăsarea butonului? Cum ați putea corecta circuitul? Ce valoare de rezistență ați alege?

d) Rezolvați punctul (b) folosind întreruperi.

e) Să se determine timpul de răspuns la modificarea valorii pe pinul B3 în cele două variante (monitorizare, întreruperi). Se dă $fclock$ = 1 MHz (16F84).

4. Să se scrie codul în limbaj de asamblare și în C pentru a realiza următoarele operații:

a) Se generează continuu pe pinul B2 impulsuri de durata – măsurată în cicli mașină - egală cu valoarea numărului prezentat la portul B pe pinii B4-B7 și cu factor de umplere 1/2.

b) La pinul B3 se conectează capătul unui rezistor care, printr-un comutator; este conectat la +5V. La închiderea comutatorului, a cărui poziție normală este „circuit deschis", microcontrolerul trebuie să genereze impulsuri cu factor de umplere variabil, începând de la factorul 1/10 până la factorul 9/10, cu pas de 2/10, în buclă infinită (chiar dacă butonul nu mai este apăsat). Rezolvați prin monitorizarea în buclă a pinului B3.

1. să se scrie codul in limbaj de asamblare și in C pentru a realize urmatoarele operații:

a) Se preia de la portul B0-B7 un număr binar

b) Se determina daca numărul este par sau impar.

c) Dacă numărul este par, se aplică operatia SAU intre număr și constanta zecimala 9 iar la rezultat se aduna constanta zecimala 5.

d) Rezultatul se pune in memorie la o locatie in bank-ul al doilea.

e) Daca numărul citit este impar, se împarte la 2, iar rezultatul se pune in memorie.

f) Estimați durata executării programului daca fclock = 1 MHz (16F84), pentru cazul numărului impar.

2. să se scrie codul in limbaj de asamblare și in C pentru a realiza următoarele operatii:

a) Se generează in bucla infinita pe pinul B2 un tren de impulsuri cu durata de T1 = 1ms (in "1"logic). Perioada T intre doua impulsuri consecutive este de 100 ms.

b) La apăsarea unui buton (atașat pinului B0) se oprește generarea de impulsuri, respective la apăsarea din nou a butonului de repornește generarea semnalului.

c) să se determine timpul de răspuns de la modificarea pinului RB0 și pana la oprirea/startarea generarii trenului de impulsuri in cele doua variante (monitorizare, întreruperi). fclock = 4 MHz (16F84).

3. Sa se scrie codul in limbaj de asamblare și in C pentru a realiza următoarele operații:

a) Se generează un semnal PWM cu factor de umplere inițial de 50%. Se repetă generarea semnalului in buclă infinită.
b) Pe pinii B4, B5 sunt doua butoane. Daca se apăsă aceste butoane factorul de umplere al semnalului PWM creste, respectiv scade fără a putea lua valori in afara intervalului [0-100]%. Se va rezolva acest punct in două variante: prin monitorizarea in buclă a pinului B3 și prin întreruperi.

c) să se determine timpul de răspuns de la apăsarea unuia din cei doi pini B4, B5 și pana la modificarea valorii registrului ce stochează valoarea factorului de umplere a semnalului PWM in cele doua variante (monitorizare, întreruperi). fclock = 4 MHz (16F84).

6. A) Să se deseneze schema de principiu pentru un procesor cu instrucțiuni având *opcodul* de 12 biți și o memorie de program (MdP) de 1 k-cuvinte; anume, se va desena la nivel de blocuri stiva, numărătorul de program (NdP) și magistralele care le conectează pe acestea, precizând pe desen dimensiunea cuvintelor și magistralelor. B) Procesorul are ALU de 8 biți. Credeți că procesorul este de tip RISC sau CISC? Justificați pe scurt răspunsul.
7. Duratele aproximative ale funcționarii blocurilor unui uC sunt: citire MdP – max 10 ns; decodificare instrucțiuni max 6 ns; operare ALU max 8 ns; adresare și scriere MdD - max 15 ns. Propuneți un mod de lucru *pipeline* rezonabil pentru aceste date, precizând durata unei faze.
8. a) Precizați o instrucțiune la/după a cărei citire se transferă informație pe magistrala dintre numărătorul de program (NdP) și stivă. b) Precizați o instrucțiune pentru care transferul este dinspre NdP spre stivă. (c) Câte cuvinte se transferă, câți biți au (ce determină numărul de biți), unde sunt puse pe stivă și la ce sunt/este util(e)?

9. Descrieți structura hard (NdP, MdP, stiva, magistrale aferente, ALU…) la uP EASI 20XX caracterizat prin: a) MdP = 1 k-cuvinte; b) set instrucțiuni RISC de 49 de instrucțiuni, cu instrucțiuni aritmetice-logice cu același format ca la 16FXX (ex., AND d, ff…f); c) MdD 4 kB organizată pe 4 file (*banck*-uri); d) procesor pe 16 biți; e) permite până la 12 bucle imbricate în program (sau combinații de bucle și întreruperi). Procesorul este CISC; RISC; Von Neumann; Harvard; Harvard extinsă? (justificați)
10. Un microcontroler executa (doar) 8 operații, dintre care doua folosesc constante, iar patru accesează valori din memoria de date. Memoria de date este organizata pe cuvinte de 12 biți (ne-uzual), ca și dimensiunea registrilor ALU, iar capacitatea completă a memoriei de date este de 1 k-cuvânt (1024 cuvinte de 12 biți). Codul instrucțiunilor care accesează memoria de date are doar 8 biți tip (fff…f).
    a) De câți biți avem nevoie pentru a identifica o instrucțiune?
    b) Câți biți sunt necesari pentru codul instrucțiunilor? De ce?
    c) Câți biți are un cuvânt din memoria de program?
    d) Este necesar să organizam memoria de date pe bankuri? De ce?
    e) Cate cuvinte include un bank?
    f) Cate bancuri are memoria de date?
    g) Care este legătura dintre numărul de biți ai numărătorului de program și cele de mai sus? Explicați.
    h) Explicați legătura (relația) dintre numărul de biți ai registrilor ALU și consumul de putere al ALU.
    i) Este vreo legătura intre tipul de capsulă folosită pentru un microprocesor și frecvența maximă de tact a procesorului? Explicați.

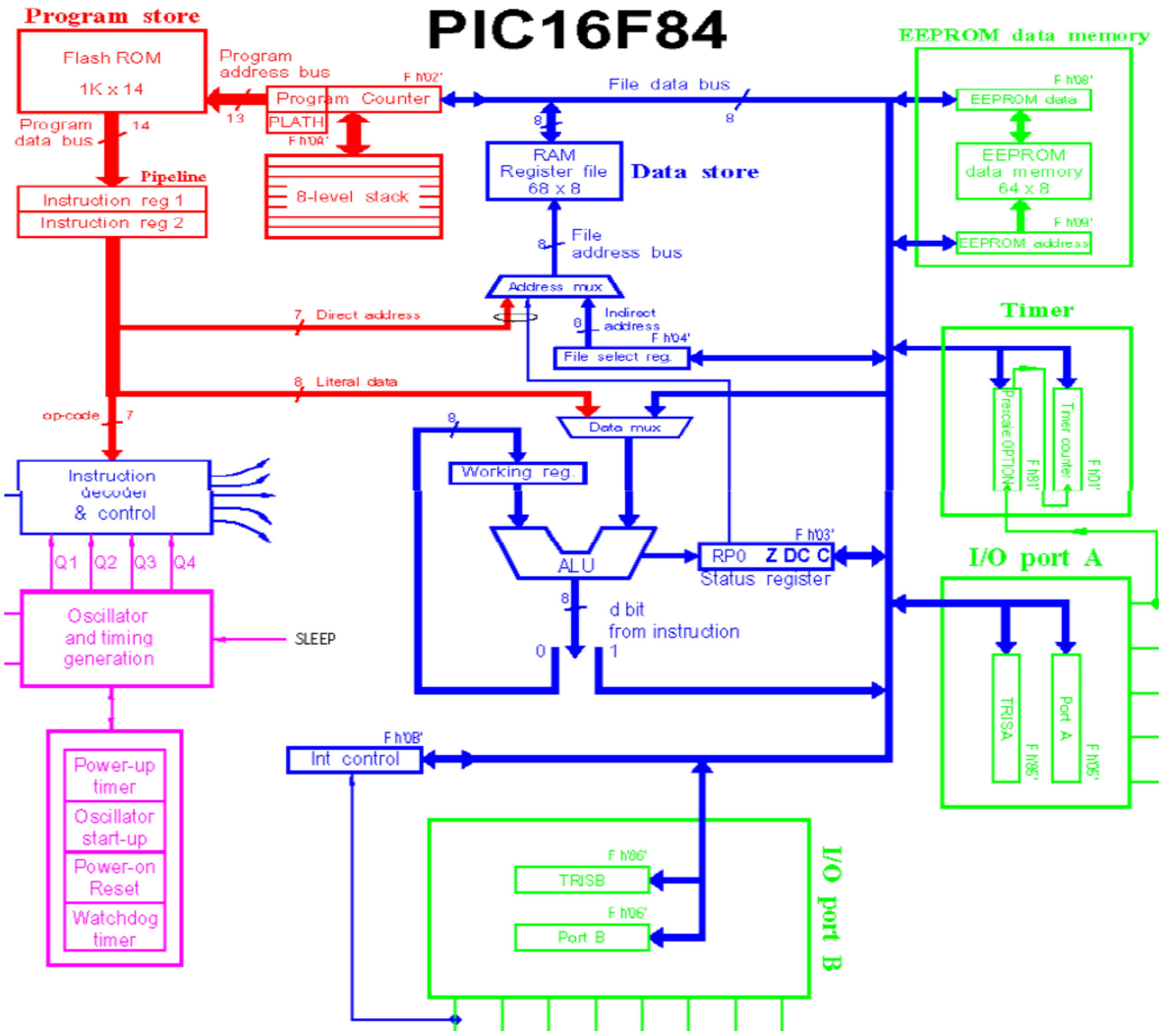
    Întrebările (h) și (i) sunt suplimentare.

4. Sa se explice cum se pot determina numerele de cuvinte necesare pe o stivă?
5. Funcție de numărul de salturi sau bucle imbricate in programele lucrare?
6. Care este magistrala de adresare a stivei (de program)?
   Nu are magistrala de adresare.
7. Cum se adresează stiva?

Nu se adresează, e LIFO.

8. Registrii pentru memoria de program au cați biți – cum se determină?

9. Desenați cât mai complet schema bloc a unui microcontroler.
10. Explicați cum se corelează dimensiunea (numărul de cuvinte al) stivei cu numărul de bucle imbricate care pot fi realizate intr-un program pe microsistemul respectiv.
11. Explicați cu cine și cum se corelează numărul de biți ai unui cuvânt al stivei.
12. Scrieți, pentru microcontrolerul PIC 16F84, un fragment de program care :
    a) accepta o întrerupere **prioritara** pe B0, iar la sosirea întreruperii respective:
    b) salvează acumulatorul și conținutul registrului de stare
    c) apoi aduna conținutul acumulatorului cu 1000111, împarte rezultatul la 4, apoi
    d) pune rezultatul pe portul B setat ca ieșire.


PIC16F84
Program store
Flash ROM
1K x 14
Program address bus
13
Program Counter
PCLATH
F h'02'
F h'0A'
Program data bus
14
Pipeline
Instruction reg 1
Instruction reg 2
8-level stack
File data bus
8
RAM
Register file
68 x 8
Data store
File address bus
Address mux
7 Direct address
8 Indirect address
F h'04'
File select reg.
8 Literal data
op-code
7
Data mux
Working reg.
ALU
F h'03'
RP0 Z DC C
Status register
d bit
from instruction
Instruction decoder & control
Q1 Q2 Q3 Q4
Oscillator and timing generation
SLEEP
Power-up timer
Oscillator start-up
Power-on Reset
Watchdog timer
F h'0B'
Int control
TRISB
F h'06'
Port B
F h'06'
I/O port B
EEPROM data memory
EEPROM data
F h'08'
EEPROM data memory 64 x 8
EEPROM address
F h'09'
Timer
Prescale OPTION
F h'81'
Timer counter
F h'01'
I/O port A
TRISA
F h'85'
Port A
F h'05'