Folosirea limbajului C în programarea microcontrolerelor

Asistent Maria Radu
Universitatea "Tibiscus" din Timişoara

ABSTRACT: The early name for a microcontroller was *microcomputer,* but a microcontroller differs from a microprocessor in several important ways. A microprocessor was simply the "heart" of a computer. To put a microprocessor into use, the designer required memory, peripheral chips, and serial and parallel ports to make a completely functional computer. By contrast, the microcomputer was designed to be a complete computer on a single chip.

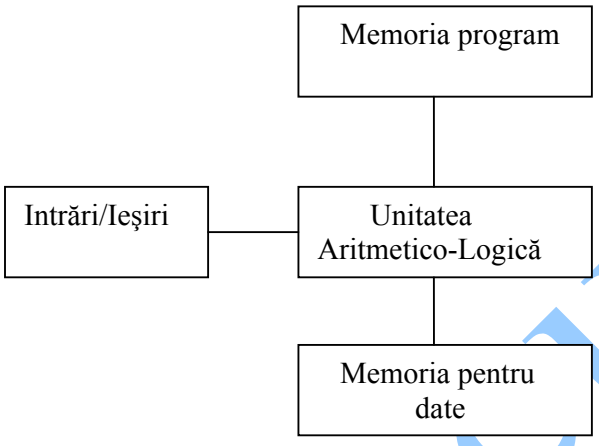
1 Introducere

Microcontrolerele au apărut odată cu dezvoltarea circuitelor integrale când s-a făcut posibilă înmagazinarea a sute de mii de tranzistoare într-un singur cip. Astfel au început să se construiască microprocesoare, iar prin adăugarea memoriilor externe, a dispozitivelor de intrare/ieşire, etc s-au constituit primele calculatoare. Astăzi, utilizarea microcontrolerelor s-a extins la un nivel foarte înalt. Într-o casă modernă, microcontrolerele sunt pretutindeni, de la maşina de spălat la cuptorul cu microunde şi mai departe.

Deşi la început microcontrolerele au fost denumite microcomputere, un microcontroler diferă de un calculator în câteva moduri importante, marea diferența constă în faptul ca un microcontroler este construit a fi un computer dintr-un singur cip. Memoria externă, perifericele şi celelalte lucruri necesare unui computer pentru a fi funcțional sunt integrate în acest cip, tocmai de aceea au consum redus şi pot executa aplicații complexe de achiziție, monitorizare şi control în timp real.

2 Arhitectura unui microcontroler

Arhitectura generală a unui microcontroler este următoarea:



iar această arhitectură diferă de la un microcontroler la altul în funcție de mai mulți factori (producătorul sau funcționalitatea microcontrolerului, de tipul lui, etc).

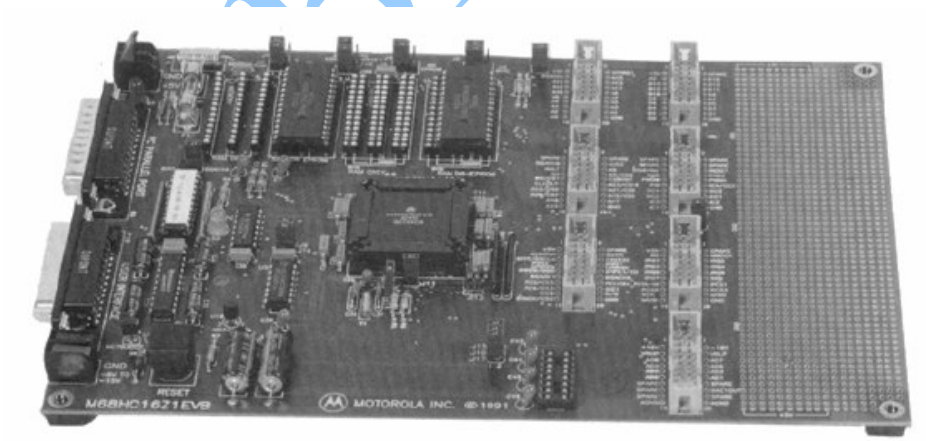


fig.2 Exemplu de microcontroler

Un microcontroler execută un program dat astfel: programul este stocat în memorie iar unitatea arimetico-logică citeşte o instrucțiune din

memorie, decodează instrucțiunea citită şi o execută. La terminarea execuției unei instrucțiuni, o altă instrucțiune este transportată din memorie şi este executată. Acest procedeu este repetat până se găseşte sfârşitul programului sau dacă programul intră într-un ciclu infinit.

Mai înainte de a executa un program, trebuie să dăm viață microcontrolerului conectându-l la o sursă de tensiune, deoarece numai la conectare acesta execută un scurt control asupra sa, localizează începutul programului şi apoi începe să execute acest program.

Cum va lucra aparatul depinde de mulți parametri, cel mai important fiind priceperea dezvoltatorului de hardware, şi de experiența programatorului în obținerea maximului din aparat cu programul său.

3 Programarea microcontrolerelor

Un domeniu special în lucrul cu microcontrolerele este, deci, scrierea programelor, adică programarea microcontrolerelor. Înainte de a se hotărî în ce limbaj se va programa un microcontroler trebuie să se țină seama de cerințele privind viteza de execuție, mărimea memoriei şi timpul disponibil pentru asamblarea sa. La început, şi acum uneori, programarea microcontrolerelor se făcea numai în limbaj de asamblare, limbaj de nivel scăzut în care programarea se face lent şi greoi, dar care foloseşte spațiu mic de memorie şi dă cele mai bune rezultate când se are în vedere viteza de execuție a programului. În momentul actual se tinde să se programeze microcontrolerele în limbaje mai avansate, limbaje de nivel înalt. Primul şi cel mai utilizat ar fi limbajul C deoarece deşi este un limbaj de nivel înalt, păstrează cel mai bine contactul cu partea hardware a unui computer, contact care în mod normal se pierde la limbajele de nivel înalt. Nici un aspect al părții hardware nu poate fi ascuns programării în limbajul C, deoarece acesta are facilități pentru manipularea biților, a câmpurilor de biți, adresarea directă a memoriei, precum şi manipularea funcțiilor cu pointeri la funcții. Cu C avem toate avantajele unui limbaj uşor de înțeles, a unui limbaj standardizat, a unui limbaj în care orice programator poate înțelege programele altui programator, a unui limbaj foarte productiv. Programele în limbajul C sunt mai uşor de scris, mai uşor de înțeles, dar există şi un oarecare dezavantaj, sunt mai lente în executare decât programele în limbaj de asamblare.

Deoarece majoritatea arhitecturilor de microcontrolere sunt proiectate pentru a fi programate în limbaj de asamblare, mai întâi s-au dezvoltat mai mult arhitecturi care să fie eficiente atât pentru limbajul C, cât

şi pentru limbaje de asamblare, apoi programarea microcontrolerelor în C a devenit o certitudine.

Plecând de la limbajul C, prin modificarea semnificativă a acestui limbaj, s-au concretizat alte medii de dezvoltare software, de exemplu Dynamic C™, C6805 pentru familia de microcontrolere M68HC05, etc.

După cum am subliniat, asupra limbajului C s-au făcut unele modificări, deci unele instrucţiuni pentru microcontrolerele M68HC05 (de exemplu) nu au echivalent în limbajul C standard. Aceste instrucţiuni sunt:

- CLC sau CLC() şterge bitul carry (de transport)
- SEC sau SEC() setează bitul de transport
- CLI sau CLI() şterge indicatorul de întrerupere
- SEI sau SEI() setează indicatorul de întrerupere pe off
- NOP sau NOP() nici o operaţie
- RSP sau RSP() resetează pointerul de stivă
- STOP sau STOP() instrucţiunea STOP
- SWI sau SWI() întrerupere software
- WAIT sau WAIT() instrucţiunea WAIT

Se folosesc, de altfel, şi un set de directive pe care nu le întâlnim în limbajul C standard:

- Directive *#pragma* – C6805 utilizează aceste directive pentru a identifica caracteristicile specifice microcontrolerelor. Forma standard a unei astfel de directive este:

#pragma portxx nume_port @ adresă

unde *portxx* poate fi port de citire, de scriere sau amândouă (*portr, portw, portrw*), *nume_port* este numele utilizat in program pentru port iar simbolul @ identifică o adresă de memorie specificată prin *adresă.*

Când programăm în C, având în vedere că microcontrolerele au o memorie strict limitată, trebuie să ţinem seama de anumite reguli astfel încât memoria disponibilă să fie programată în cel mai eficient mod. Câteva dintre aceste reguli sunt următoarele:

- ori de câte ori nu este necesar să nu se definească variabile, să se definească constante cu nume cât mai clar, scris cu litere mari,
- evitarea folosirii variabilelor globale. Este foarte important ca în loc de variabile globale să se folosească variabile locale (definite într-o rutină), deoarece variabilele locale vor aloca memoria RAM doar atunci când se execută rutina în care sunt declarate variabilele în timp ce variabilele globale ocupă memoria RAM în permanenţă
- numele funcţiilor să fie cât mai explicit, iar dacă se folosesc mai multe cuvinte în acest nume, fiecare cuvânt să înceapă cu literă mare

- argumentele unei funcţii să fie transmise într-o ordine logică şi să se folosească cat mai puţine argumente. Dacă este necesară transmiterea mai multor argumente, mai bine se construieşte o structură în care să se încorporeze aceste argumente, iar funcţiei i se va transmite adresa unei variabile de tipul structurii astfel construite
- ori de câte ori avem definite structuri să se redefinească aceste structuri cu *typedef*
- definirea macrourilor în loc de funcţii deoarece acestea vor fi compilate in-line
- cod în limbaj de asamblare va fi inclus în codul în limbajul C doar dacă acea parte de cod nu poate fi scrisă în C
- în programele scrise pentru microcontrolere trebuie să se evite cât mai mult folosirea tipurilor cu semn, cu precădere se vor folosi tipurile fără semn
- se vor include cât mai multe comentarii pentru a documenta cât mai bine programul.

Bibliografie

[TVS00] **Ted Van Sickle**- *Programming Microcontrollers in C,* LLH Technology, 2000

[WWWa] PICBook, http://www.mikroelektronika.co.yu/romanian/books/

[WWWb] 02.htm,http://www.mikroelektronika.co.yu/magazine/articles/