Laborator 5

Obiectivele acestui laborator sunt familiarizarea cu mediul de dezvoltare MPLAB X IDE și rularea câtorva exemple simple de programe scrise în limbajele C și assembler folosind placa de dezvoltare Digilent chipKIT uc32.

5.1 Utilizarea mediului MPLAB X

MPLAB este un mediu de dezvoltare integrat, creat de Microchip Technology pentru aplicațiile de tip embedded al microcontrolerelor PIC și AVR. Prin mediu de dezvoltare integrat (IDE) se înțelege o aplicație software ce pune la dispoziția programatorilor instrumente pentru crearea aplicațiilor software. Printre aceste instrumente se regăsesc editorul de text, unelte pentru crearea codului obiect, rularea și depanarea acestuia.

5.1.1 Crearea unui proiect nou

Pentru a crea un proiect nou se alege din meniu intrarea
Project -> Project Wizard
Din lista de categorii se selectează *Microchip Embedded*, iar dintre proiecte *Standalone Project*

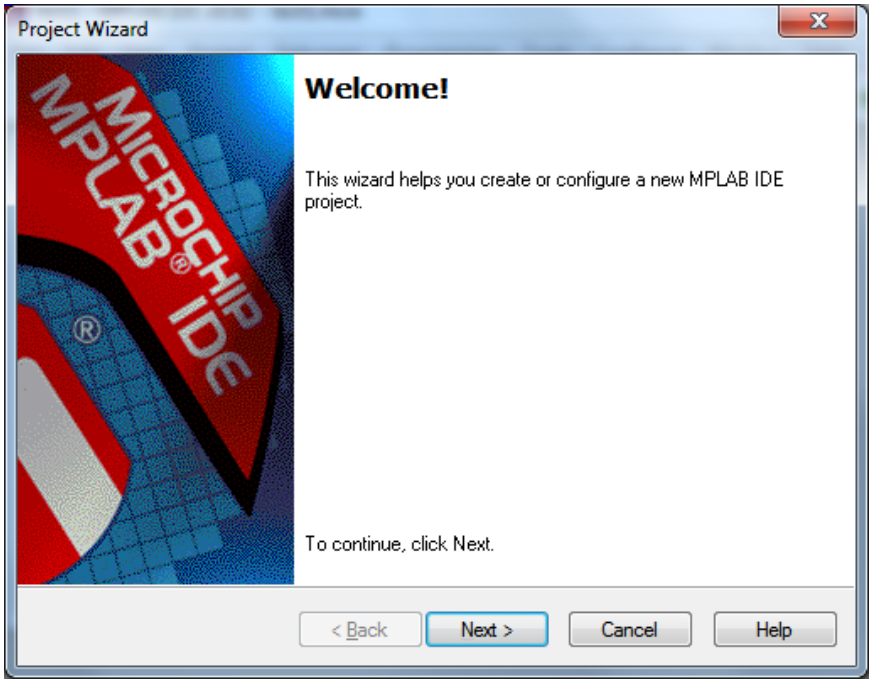


Figura 1 Fereastra New Project

Mai departe este ales dispozitivul de lucru. *Familia* aleasă este cea a microprocesoarelor PIC pe 32 de biți, iar dispozitivul este *PIC32MX340F512H*.

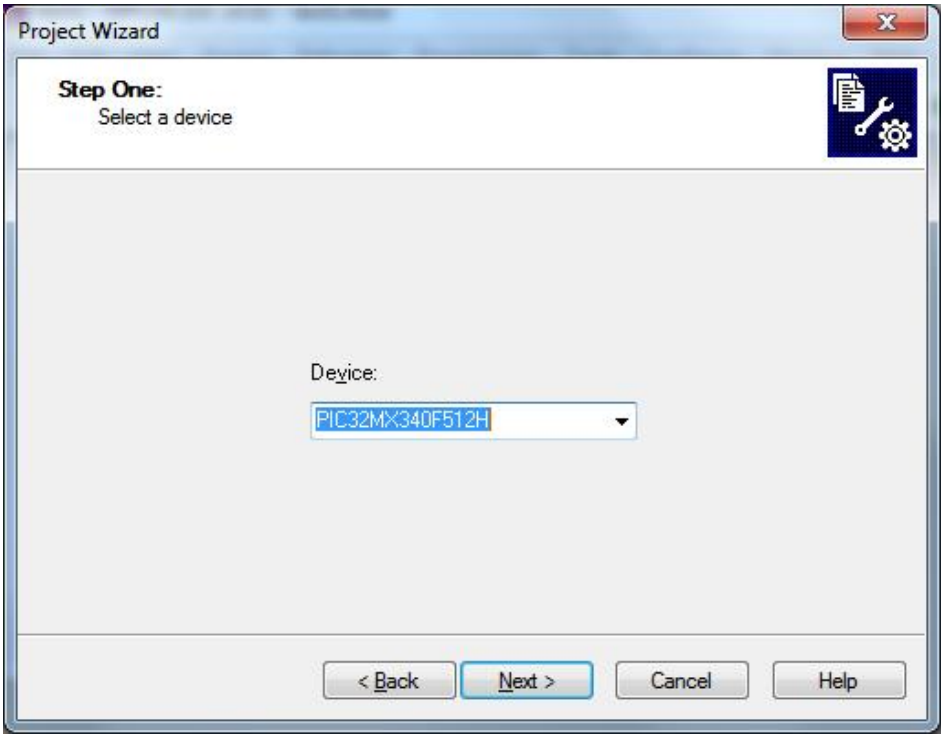
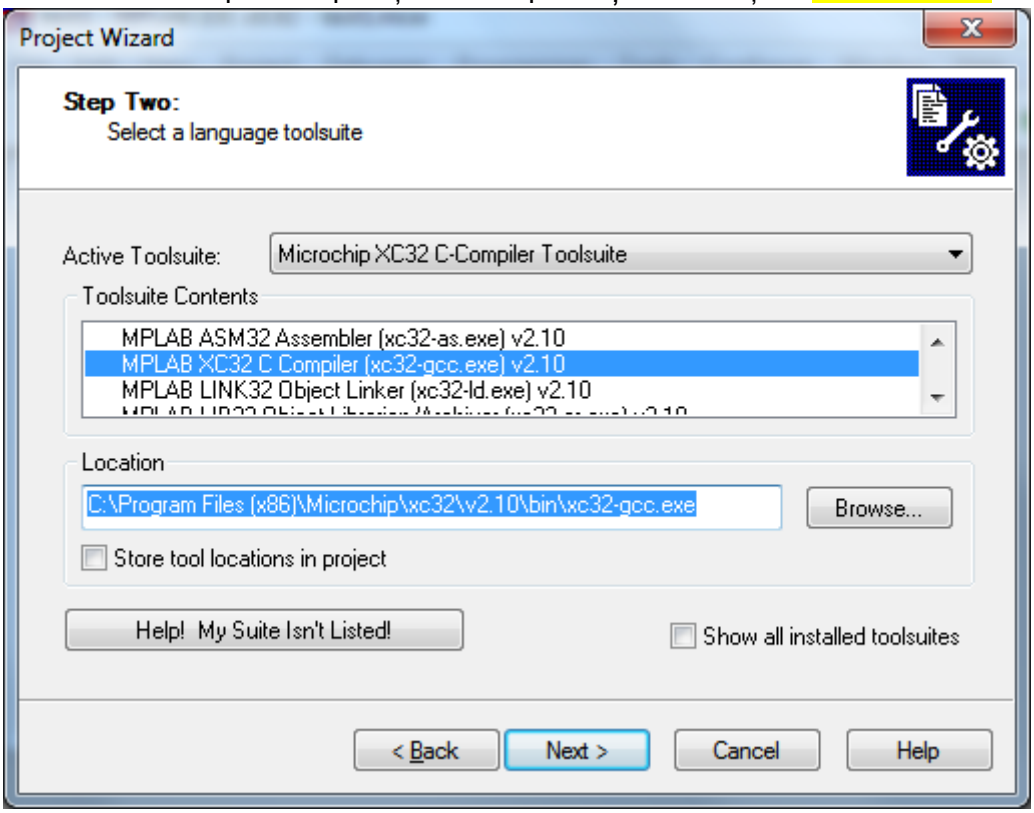


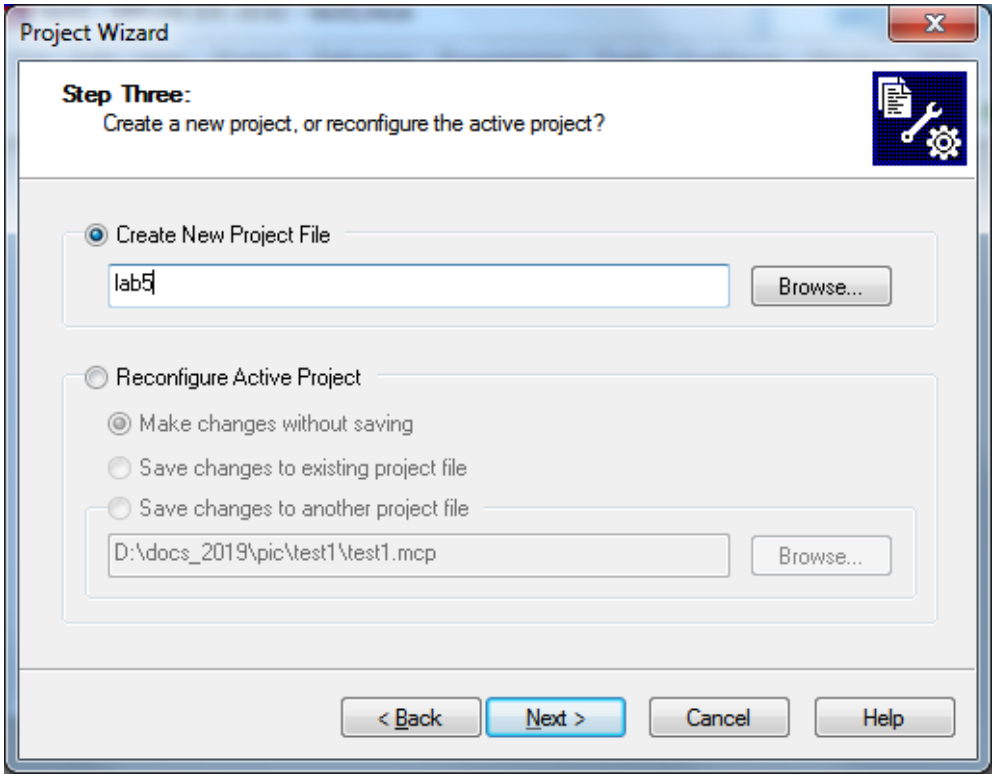
Figura 2 Fereastra New Project - Select Device

Unele medii de dezvoltare integrate includ printre instrumente și un verificator de sintaxă și un compilator. Alte medii apelează pentru acestea la soluții externe. Este și cazul MPLAB, care apelează la setul *XC32* pentru aplicația de compilare și construcție a codului obiect.



Procesul de creare al unui proiect nou se termină prin denumirea acestuia, se poate da numele *lab5* proiectului pentru pasul următor.

Se alege un nume pentru proiect și calea către directorul de lucru:



5.1.2 Crearea și adăugarea fișierelor sursa

Pentru a crea un fișier de cod ce se va adăuga proiectului, se apelează intrarea:
File -> Add new file to project
În cadrul acestui laborator, se va crea un fișier cu extensia .c.
Drept exemplu, se va studia un cod scris în limbajul C ce descrie o buclă simplă.

5.2 Testarea pe placă

5.2.1 Placa de dezvoltare chipKIT uc32

UC32 este o platformă prototip pe baza microcontrolerului PIC32MX340F512, cu un nucleu de procesor MIPS pe 32 de biți ce rulează la 80 MHz, 512 KB de memorie de program Flash și 32 KB de memorie de date SRAM. Conectarea la IDE se face prin portul USB serial avut la dispoziție. Pentru alimentare se poate folosi atât portul USB cât și o sursă de alimentare externă.

UC32 oferă 42 de pini I/O care acceptă o serie de funcții periferice, cum ar fi porturile UART, SPI, I2C și ieșiri modulate cu lățimea impulsului. Doisprezece dintre pinii I/O pot fi folosiți ca intrări analogice sau ca intrări și ieșiri digitale.

Conectori:

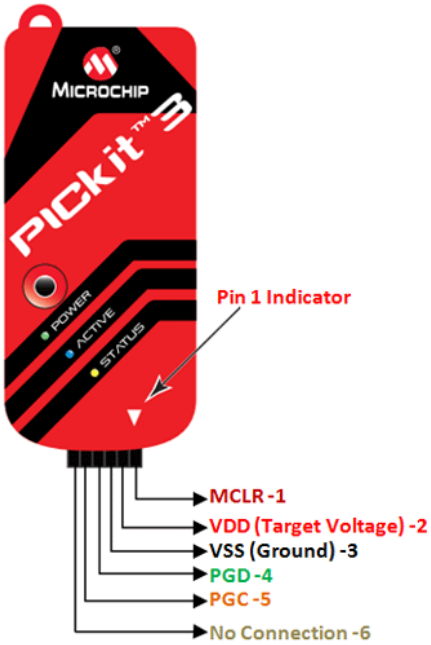
- Conector USB pentru convertor serial USB
- Conector tip bară de 5,5 x 2,1 mm utilizat pentru alimentarea uC32 de la o sursă de alimentare externă

Caracteristici:

- 512KB Flash
- 32KB RAM
- Viteza de operare de 80 MHz
- Două interfețe SPI și două interfețe I2C
- 16 canale ADC pe 10 biți, cinci ieșiri PWM

- 42 pini de I/O disponibili
- Tensiune de operare 3.3V
- 12 intrări analogice
- Conexiunea PC utilizează un cablu USB A la mini-B (nu este inclus)
- Tensiune de intrare de la 7V la 15V
- +/- Curent de 18mA DC pe pin
- Curent de funcționare tipic 75mA
- Două LED-uri pentru utilizator
- programabil prin MPIDE și Microchip MPLAB IDE

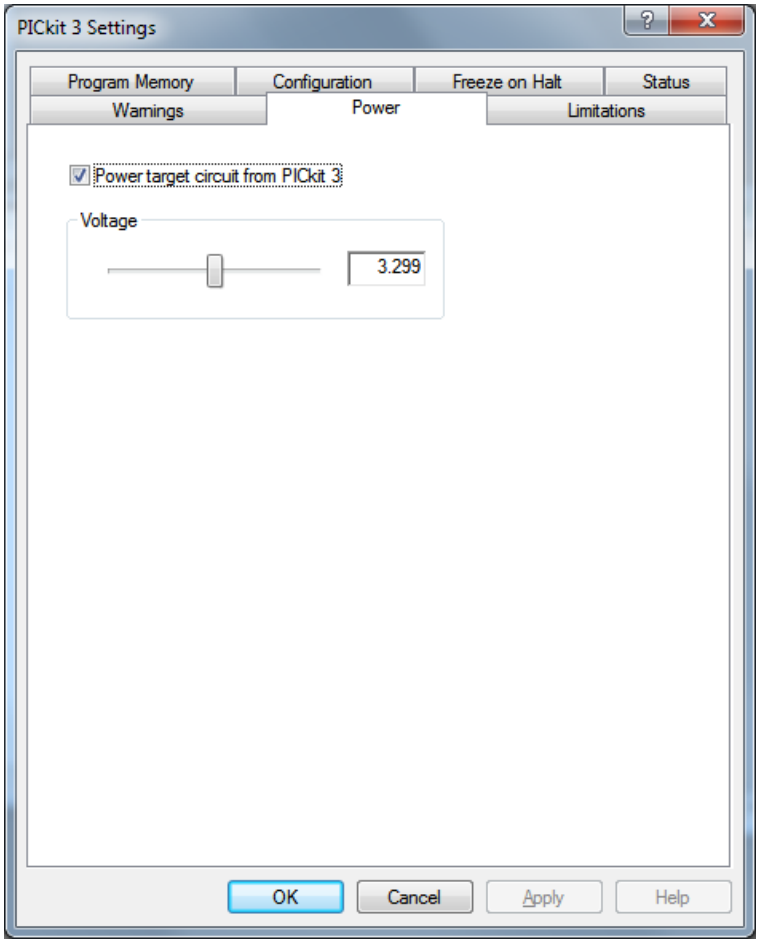
5.1.3 Programatorul / Debugger-ul Micro PiCkit 3



Prin conectorul prezentat mai sus programatorul / debugger-ul PiCkit 3 se conecteaza la conectorul JP3 al plăcii de dezvoltare uc32. Acesta permite încărcarea programelor în memoria microprocesorului, depanarea programelor și rularea acestora.
Programatorul comunica cu calculatorul prin intermediul cablului USB.

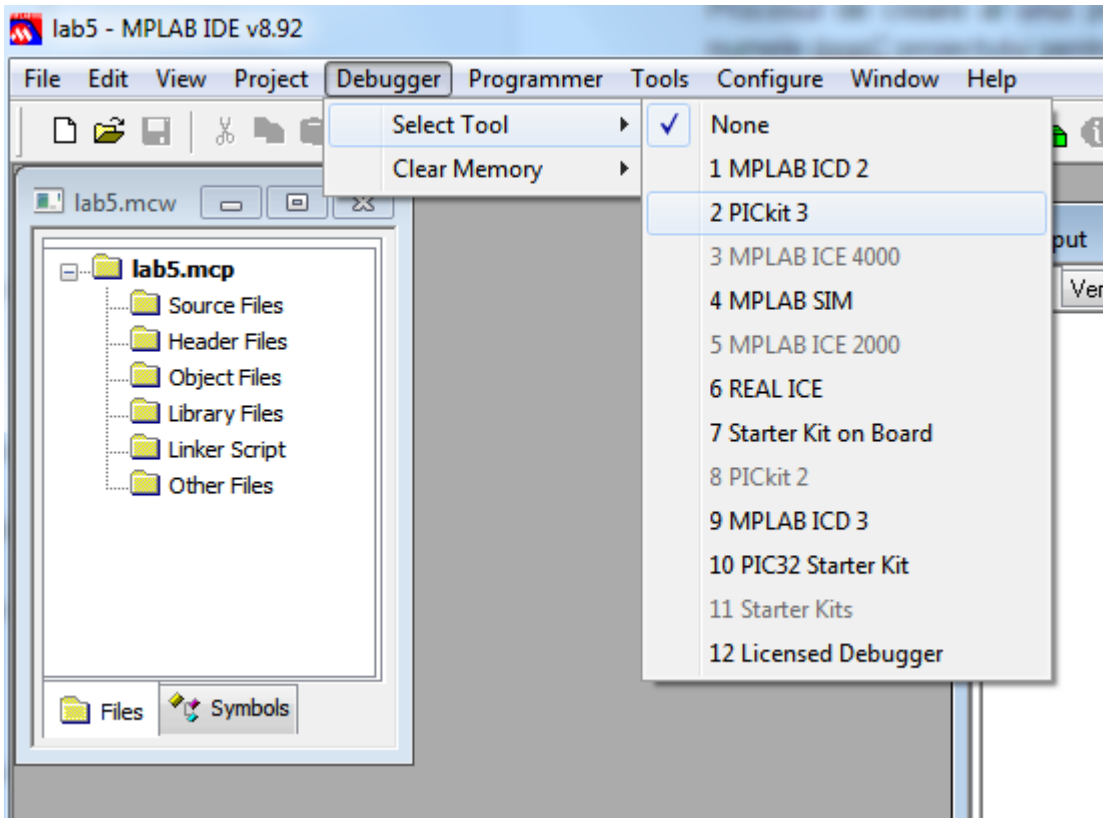
5.2.3 Alimentarea chipKIT uc32 prin intermediul programatorului

Pentru alimentarea plăcii de dezvoltare de la programator trebuie făcută setarea următoare: Debugger -> Settings -> Power și se bifeaza “Power Target Circuit from PICkit 3” ca in figura de mai jos:



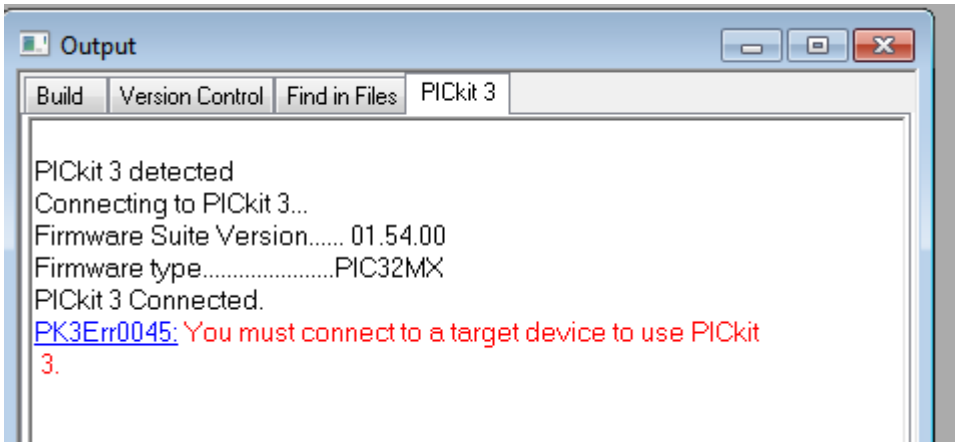
5.2.4 Încărcarea pe placă

Mai întâi se conectează programatorul la calculator si implicit la mediul de dezvoltare:



Dacă nu a fost realizata setarea de la pasul anterior, se va obține o eroare ca cea de mai jos, caz în care trebuie bifata opțiunea pentru alimentarea plăcii de dezvoltare de la programatorul PICkit 3.

:



Se compileaza codul c pentru a se obține codul ce va fi incarcat in memorie.

Pentru încărcarea pe placa, după conectarea plăcii de dezvoltare la programator și conectarea programatorului la calculator se apelează:
Debugger - > Program.
În acest moment programul a fost încărcat în memoria microprocesorului.

5.2.5 Modul debug și rularea pas cu pas

Pentru rularea pas cu pas se procedează astfel:

- se setează un breakpoint pe una dintre instrucțiuni
- se deschide codul în limbaj de asamblare, de la View -> Disassembly listing
- se începe rularea programului, Debugger -> Run. Programul se va opri în dreptul instrucțiunii la care a fost setat breakpoint-ul.
- În continuare se poate rula pas cu pas, prin apăsarea taste F7.

Tot de la tab-ul View se pot observa regiștrii microprocesorului precum și memoria acestuia.

5.3 Exerciții

- Să se realizeze un program în limbaj C care incrementează o variabila de tip int într-o bucla infinită.
- Încărcați programul de placa de dezvoltarea Arduino uc32.
- Vizualizați codul în limbaj de asamblare să răspunde-ti la următoarele întrebări:
  - **a.** Notați programul în limbaj de asamblare
  - **b.** În ce registru este stocată valoarea care se incrementează? Pe cati biti este acest registru.
  - **c.** În ce instrucțiuni este dezasamblata instrucțiunea de nivel înalt while(1). Explicați aceste instrucțiuni.
  - **d.** Care este adresa fizica la care este stocat codul în memoria program? Care este adresa fizica din memoria de date la care este stocată variabila ?
  - **e.** Care sunt adresele logice ale zonei de memorie în care este stocat programul și variabila de tip int?
  - **f.** Identificați în lista de regiștrii, registrul numărător de instrucțiuni și registru SP.

- Modificați tipul variabilei din int în long și inițializați-o cu valoarea maxima pe 32 de biți.
  - **a.** Cum este stocată variabila in memorie.
  Răspuns: 2 regiștrii a câte 32 de biți `v0` și `v1` vor fi folosiți pentru stocarea intermediară a acestei variabile; cei doi regiștrii vor fi stocați în două locații

succesive de memorie *8(s8)* și *12(s8)*, unde *s8* este registrul *fp (frame pointer)* utilizat de subrutine pentru accesul zonei de memorie (stivă) corespunzătoare apelului

```
5:                      long long var_b;
6:
7:                      var_b = 0x00000000FFFFFFFF;
9D0000E8  2402FFFF    addiu       v0,zero,-1
9D0000EC  00001821    addu        v1,zero,zero
9D0000F0  00400013    mtlo        v0
9D0000F4  00600011    mthi        v1
9D0000F8  00001012    mflo        v0
9D0000FC  00001810    mfhi        v1
9D000100  AFC20008    sw          v0,8(s8)
9D000104  AFC3000C    sw          v1,12(s8)
```

b. Care sunt instrucțiunile în limbaj de asamblare pentru incrementarea valorii de tipul long.

Răspuns:

- Instrucțiunea *lw (load word)* va fi folosită pentru a aduce partea inferioară a variabilei long în registrul *a0*, iar partea superioară în *a1*
- Instrucțiunea *addu (add unsigned)* va fi folosită pentru incrementarea părții inferioare (`addu v0,a0,a2`) cu 1
- Depășirea valorii maxime pe 32 de biți se poate afla verificând dacă noua valoare rezultată (`v0`) este mai mică decât fosta valoare (`a0`). Pentru acest lucru se folosește instrucțiunea *sltu (set less than unsigned)* ce va atribui registrului `t0` valoarea 1 în caz că s-a produs o depășire (`v0<a0`)
- Noile valori vor fi stocate înapoi în memorie folosind instrucțiunea *sw (store word)*

```
10:                         var_b += 1;
9D000114  8FC40008    lw          a0,8(s8)
9D000118  8FC5000C    lw          a1,12(s8)
9D00011C  24060001    addiu       a2,zero,1
9D000120  00003821    addu        a3,zero,zero
9D000124  00861021    addu        v0,a0,a2
9D000128  0044402B    sltu        t0,v0,a0
9D00012C  00A71821    addu        v1,a1,a3
9D000130  01032021    addu        a0,t0,v1
9D000134  00801821    addu        v1,a0,zero
9D000138  AFC20008    sw          v0,8(s8)
9D00013C  AFC3000C    sw          v1,12(s8)
```