Задание

за курсов проект по дисциплините „Микропроцесори“ и „МПТ“

Дадена е програма на Асемблер за сортиране във възходящ ред на 32-битови думи (тип `int32_t`), тестова програма на C++ и файл за автоматична компилация (`Makefile`). Задачата е да се направят **промени** в сортиращата програма за нов тип данни (8 или 16-битов, с или без знак) и евентуално за промяна на реда от възходящ в низходящ. Това изглежда лесно, но може да се постигне само след **разбиране** на понятия като „индекс“, „адрес“ и „адресно отместване“, както и на разликата между числата с и без знак.

Номерът на заданието (в 16-ична бройна система!) е даден в **първата** колонка на таблицата. Първата цифра на този номер съвпада с последната цифра на IP-адреса на сървъра във **втората** колонка, където е дадена командата за свързване към този сървър. Тя се състои от име на командата, име на потребителя и въпросния IP-адрес. Паролата за достъп ще ви бъде дадена от асистента, но при първото вписване ще трябва да си я смените. **Третата** колонка е командата, която трябва да изпълните, за да получите файл „`sort.s`“, върху който ще правите промените. В него пише кой метод се използва за сортиране. Тези прости методи работят бавно при голям масив. Повече за тези методи вж на *http://en.wikipedia.org/wiki/Insertion_sort* и *http://en.wikipedia.org/wiki/Gnome_sort* . В **четвъртата** колонка е дадена командата за компилация на програмата. Естествено, думата „размер“ се заменя с число, означаващо броя на елементите. Това число трябва да е поне 2, но достатъчно малко (най-много няколкодесетки елемента), за да може да се види дали сортирането работи. Нужният ред за сортиране е даден в **петата** колонка.

За да можете да направите необходимите промени, трябва да знаете къде да ги направите. Първо трябва да разберете каква е **архитектурата** на съответната машина с командата „`uname -m`“. След това трябва да се запознаете с **командите**, които трябва да промените – на *http://194.141.30.37/* , където са дадени описания на системите машинни команди за всяка от използваните архитектури, както и кратки таблици на командите (в *http://194.141.30.37/ISA.pdf*). Анализирайки командите една по една, може да се разбере **как работи** програмата. Общо взето, за промяна на **размера** на елемента трябва да се промени мащабът, по който се умножава индексът, за да се получи адресно отместване, или да се промени нарастъкът на адреса. Тези числа винаги са равни на дължината на елемента на масива в байтове. За промяна на **типа** му от знаков в беззнаков или на реда на сортирането пък най-често трябва да се смени командата за сравнение или за преход.

Файлът „`sort.s`“ може да се редактира с редактор „`vi`“ (по-мощен, но необичаен за несвикналите с режимите му) или по-подходящия за начеващи „`nano`“. След успешна компилация програмата може да се пусне например с командата „`env PATH= sortst`“.

Както се вижда, всъщност целта на този курсов проект е изграждането на умения за **самостоятелно разбиране** на системите от машинни команди на микропроцесорни архитектури, които вече са изучавани (кои повече, кои по-малко), но недостатъчно, за да може това да стане без допълнително четене. Всъщност това бъдещият инженер ще трябва да прави през **целия** си професионален живот – **сам да учи нови неща**, за което сега му се дава само начален тласък. Конкретно, по-доброто разбиране на работата на процесора се отплаща стократно по-нататък, когато се наложи да се откриват и оправят грешки в програми или да се повишава производителността им.

Л. Георгиев, юни 2020 г.

40	ssh student0@194.141.30.34	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	възходящ ред
41	ssh student1@194.141.30.34	cp stupdsrt.S sort.S		
42	ssh student2@194.141.30.34	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
43	ssh student3@194.141.30.34	cp stupdsrt.S sort.S		
44	ssh student4@194.141.30.34	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
45	ssh student5@194.141.30.34	cp stupdsrt.S sort.S		
46	ssh student6@194.141.30.34	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
47	ssh student7@194.141.30.34	cp stupdsrt.S sort.S		
48	ssh student8@194.141.30.34	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	низходящ ред
49	ssh student9@194.141.30.34	cp stupdsrt.S sort.S		
4A	ssh studentA@194.141.30.34	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
4B	ssh studentB@194.141.30.34	cp stupdsrt.S sort.S		
4C	ssh studentC@194.141.30.34	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
4D	ssh studentD@194.141.30.34	cp stupdsrt.S sort.S		
4E	ssh studentE@194.141.30.34	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
4F	ssh studentF@194.141.30.34	cp stupdsrt.S sort.S		
50	ssh student0@194.141.30.35	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	възходящ ред
51	ssh student1@194.141.30.35	cp stupdsrt.S sort.S		
52	ssh student2@194.141.30.35	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
53	ssh student3@194.141.30.35	cp stupdsrt.S sort.S		
54	ssh student4@194.141.30.35	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
55	ssh student5@194.141.30.35	cp stupdsrt.S sort.S		
56	ssh student6@194.141.30.35	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
57	ssh student7@194.141.30.35	cp stupdsrt.S sort.S		
58	ssh student8@194.141.30.35	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	низходящ ред
59	ssh student9@194.141.30.35	cp stupdsrt.S sort.S		
5A	ssh studentA@194.141.30.35	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
5B	ssh studentB@194.141.30.35	cp stupdsrt.S sort.S		
5C	ssh studentC@194.141.30.35	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
5D	ssh studentD@194.141.30.35	cp stupdsrt.S sort.S		
5E	ssh studentE@194.141.30.35	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
5F	ssh studentF@194.141.30.35	cp stupdsrt.S sort.S		
60	ssh student0@194.141.30.36	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	възходящ ред
61	ssh student1@194.141.30.36	cp stupdsrt.S sort.S		
62	ssh student2@194.141.30.36	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
63	ssh student3@194.141.30.36	cp stupdsrt.S sort.S		
64	ssh student4@194.141.30.36	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
65	ssh student5@194.141.30.36	cp stupdsrt.S sort.S		
66	ssh student6@194.141.30.36	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
67	ssh student7@194.141.30.36	cp stupdsrt.S sort.S		
68	ssh student8@194.141.30.36	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	низходящ ред
69	ssh student9@194.141.30.36	cp stupdsrt.S sort.S		
6A	ssh studentA@194.141.30.36	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
6B	ssh studentB@194.141.30.36	cp stupdsrt.S sort.S		
6C	ssh studentC@194.141.30.36	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
6D	ssh studentD@194.141.30.36	cp stupdsrt.S sort.S		
6E	ssh studentE@194.141.30.36	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
6F	ssh studentF@194.141.30.36	cp stupdsrt.S sort.S		
70	ssh student0@194.141.30.37	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	възходящ ред
71	ssh student1@194.141.30.37	cp stupdsrt.S sort.S		
72	ssh student2@194.141.30.37	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
73	ssh student3@194.141.30.37	cp stupdsrt.S sort.S		
74	ssh student4@194.141.30.37	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
75	ssh student5@194.141.30.37	cp stupdsrt.S sort.S		
76	ssh student6@194.141.30.37	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
77	ssh student7@194.141.30.37	cp stupdsrt.S sort.S		
78	ssh student8@194.141.30.37	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	низходящ ред
79	ssh student9@194.141.30.37	cp stupdsrt.S sort.S		
7A	ssh studentA@194.141.30.37	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
7B	ssh studentB@194.141.30.37	cp stupdsrt.S sort.S		
7C	ssh studentC@194.141.30.37	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
7D	ssh studentD@194.141.30.37	cp stupdsrt.S sort.S		
7E	ssh studentE@194.141.30.37	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
7F	ssh studentF@194.141.30.37	cp stupdsrt.S sort.S		
80	ssh student0@194.141.30.38	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	възходящ ред
81	ssh student1@194.141.30.38	cp stupdsrt.S sort.S		
82	ssh student2@194.141.30.38	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
83	ssh student3@194.141.30.38	cp stupdsrt.S sort.S		
84	ssh student4@194.141.30.38	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
85	ssh student5@194.141.30.38	cp stupdsrt.S sort.S		
86	ssh student6@194.141.30.38	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
87	ssh student7@194.141.30.38	cp stupdsrt.S sort.S		
88	ssh student8@194.141.30.38	cp inssort.S sort.S	make TYPE=int8_t SIZE=размер	низходящ ред
89	ssh student9@194.141.30.38	cp stupdsrt.S sort.S		
8A	ssh studentA@194.141.30.38	cp inssort.S sort.S	make TYPE=uint8_t SIZE=размер	
8B	ssh studentB@194.141.30.38	cp stupdsrt.S sort.S		
8C	ssh studentC@194.141.30.38	cp inssort.S sort.S	make TYPE=int16_t SIZE=размер	
8D	ssh studentD@194.141.30.38	cp stupdsrt.S sort.S		
8E	ssh studentE@194.141.30.38	cp inssort.S sort.S	make TYPE=uint16_t SIZE=размер	
8F	ssh studentF@194.141.30.38	cp stupdsrt.S sort.S		

Ето например промените, които трябва да се направят за архитектура *PA-RISC* (не съвпада с никоя от архитектурите на нашите машини) за тип `uint16_t` и низходящ ред:

```
*** inssort.S   2020-07-01 10:28:18 +0100
--- sort.S      2020-07-01 14:29:08 +0100
***************
*** 143,155 ****
        COPY    %R24,%R23   ;   for (int j = i  // Изпълнява се преди „COMBF“!
  L2:   ADDI    -1,%R23,%R23;                   - 1;         --j)
        COMIBF,<= 0,%R23,L3 ;                       j >= 0;
!       SH2ADDL %R23,%R26,%R20;Адрес на toSort[j], изпълнява се преди „COMBF“!
!       LDWS    0(%R20),%R21; toSort[j]
!       LDWS    4(%R20),%R22; toSort[j+1]
!       COMBF,<,n %R22,%R21,L3;При пре- if (toSort[j+1] < toSort[j])
!       STWS    %R21,4(%R20);<-ход се анулира!  swap(toSort[j], toSort[j+1]);
        B       L2          ; Край на вътрешния цикъл
!       STWS    %R22,0(%R20); Изпълнява се преди „B“!
  L3:   ADDIB,TR 1,%R24,L1  ;           else break;         ++i)
        NOP                 ;\_ Край на външния цикъл
  L4:   BV      0(%R2)
--- 143,155 ----
        COPY    %R24,%R23   ;   for (int j = i  // Изпълнява се преди „COMBF“!
  L2:   ADDI    -1,%R23,%R23;                   - 1;         --j)
        COMIBF,<= 0,%R23,L3 ;                       j >= 0;
!       SH1ADDL %R23,%R26,%R20;Адрес на toSort[j], изпълнява се преди „COMBF“!
!       LDHS    0(%R20),%R21; toSort[j]
!       LDHS    2(%R20),%R22; toSort[j+1]
!       COMBF,<<,n %R21,%R22,L3;При пре-if (toSort[j] < toSort[j+1])
!       STHS    %R21,2(%R20);<-ход се анулира!  swap(toSort[j], toSort[j+1]);
        B       L2          ; Край на вътрешния цикъл
!       STHS    %R22,0(%R20); Изпълнява се преди „B“!
  L3:   ADDIB,TR 1,%R24,L1  ;           else break;         ++i)
        NOP                 ;\_ Край на външния цикъл
  L4:   BV      0(%R2)

*** stupdsrt.S  2020-06-30 20:14:59 +0100
--- sort.S      2020-07-01 14:13:22 +0100
***************
*** 115,131 ****

  /* начален адрес – в %r26, брой елементи (поне 2) – в %r25 */

! sort: ADDI    -1,%R25,%R25    ;--n            void StupidSort(int a[], int n)
!       ADDI    4,%R26,%R22     ;нач.адрес+4    {
!       SH2ADDL %R25,%R26,%R25  ;краен адрес        int tmp, i = 0;
! L1:   LDWS    0(%R26),%R23    ;*a                 do {
!       LDWS    4(%R26),%R24    ;***a ___
!       COMBT,< %R23,%R24,L2    ;       \               if (a[i] > a[i+1]) {
!       ADDI    4,%R26,%R26     ;изпълнява\ce преди COMBT!  tmp = a[i+1];
!       STWS    %R24,-4(%R26)   ;          \                a[i+1] = a[i];
        COMBT,<<= %R26,%R22,L2  ;           \               a[i] = tmp;
!       STWS    %R23,0(%R26)    ;изпълнява се\преди COMBT!  if (i)
!       ADDI    -8,%R26,%R26    ;             \                 i--;
  L2:   COMBT,<< %R26,%R25,L1   ;              \_____  } else i++;
        NOP                     ;                   } while (i < n - 1);
        BV      0(%R2)          ;               }
--- 115,131 ----

  /* начален адрес – в %r26, брой елементи (поне 2) – в %r25 */

! sort: ADDI    -1,%R25,%R25    ;--n            void StupidSort(uint16_t a[], int n)
!       ADDI    2,%R26,%R22     ;нач.адрес+2    {
!       SH1ADDL %R25,%R26,%R25  ;краен адрес        uint16_t tmp; int i = 0;
! L1:   LDHS    0(%R26),%R23    ;*a                 do {
!       LDHS    2(%R26),%R24    ;***a ___
!       COMBT,<< %R24,%R23,L2   ;       \               if (a[i+1] > a[i]) {
!       ADDI    2,%R26,%R26     ;изпълнява\ce преди COMBT!  tmp = a[i+1];
!       STHS    %R24,-2(%R26)   ;          \                a[i+1] = a[i];
        COMBT,<<= %R26,%R22,L2  ;           \               a[i] = tmp;
!       STHS    %R23,0(%R26)    ;изпълнява се\преди COMBT!  if (i)
!       ADDI    -4,%R26,%R26    ;             \                 i--;
  L2:   COMBT,<< %R26,%R25,L1   ;              \_____  } else i++;
        NOP                     ;                   } while (i < n - 1);
        BV      0(%R2)          ;               }
```

Аналогичните промени (пак за тип `uint16_t` и низходящ ред) за архитектура *VAX*:

```
*** sort.S      2020-07-16 17:08:15.000000000 +0300
--- sort.S      2020-09-13 19:25:31.000000000 +0300
***************
*** 27,37 ****
  L1:   ACBL    R1,$0,8(AP),L4#          i < toSort.size();
        SUBL3   $1,R1,R2#       for (int j = i - 1;
  L2:   BLSS    L3      #                           i >= 0;
!       CMPL    4(R0)[R2],(R0)[R2]#     if (toSort[j+1] < toSort[j])
!       BGEQ    L3      #___________
!       MOVL    4(R0)[R2],R3#       \           swap(toSort[j], toSort[j+1]);
!       MOVL    (R0)[R2],4(R0)[R2]#/*\*/
!       MOVL    R3,(R0)[R2]#          \_else break;
        SOBGEQ  R2,L2   # Край на вътрешния цикъл           --j)
  L3:   AOBLEQ  8(AP),R1,L1#Край на външния цикъл           ++i)
  L4:   RET
--- 27,37 ----
  L1:   ACBL    R1,$0,8(AP),L4#          i < toSort.size();
        SUBL3   $1,R1,R2#       for (int j = i - 1;
  L2:   BLSS    L3      #                           i >= 0;
!       CMPW    2(R0)[R2],(R0)[R2]#     if (toSort[j+1] > toSort[j])
!       BLEQU   L3      #___________
!       MOVW    2(R0)[R2],R3#       \           swap(toSort[j], toSort[j+1]);
!       MOVW    (R0)[R2],2(R0)[R2]#/*\*/
!       MOVW    R3,(R0)[R2]#          \_else break;
        SOBGEQ  R2,L2   # Край на вътрешния цикъл           --j)
  L3:   AOBLEQ  8(AP),R1,L1#Край на външния цикъл           ++i)
  L4:   RET

*** stupdsrt.S  2020-09-13 21:58:17.000000000 +0300
--- sort.S      2020-09-13 21:57:43.000000000 +0300
***************
*** 14,27 ****

        .global _sort
  _sort:
! sort: .word   0x4     #R2             void StupidSort(int a[], int n)
        MOVL    4(AP),R0              # {
!       MOVL    $1,R1                 #       int tmp, i = 1;
! L1:   CMPL    -4(R0)[R1],(R0)[R1]   #       do {
!       BLEQ    L2                    #               if (a[i-1] > a[i]) {
!       MOVL    (R0)[R1],R2           #                       tmp = a[i];
!       MOVL    -4(R0)[R1],(R0)[R1]   #                       a[i] = a[i-1];
!       MOVL    R2,-4(R0)[R1]         #                       a[i-1] = tmp;
        ACBL    $1,$0,R1,L2           #_______                if (i > 1) i--;
        SUBL2   $2,R1                 #_/  __         } else i++;
  L2:   AOBLSS  8(AP),R1,L1           #___/__ } while (i < n);
--- 14,27 ----

        .global _sort
  _sort:
! sort: .word   0x4     #R2             void StupidSort(uint16_t a[], int n)
        MOVL    4(AP),R0              # {
!       MOVL    $1,R1                 #       uint16_t tmp, i = 1;
! L1:   CMPW    -2(R0)[R1],(R0)[R1]   #       do {
!       BGEQU   L2                    #               if (a[i-1] < a[i]) {
!       MOVW    (R0)[R1],R2           #                       tmp = a[i];
!       MOVW    -2(R0)[R1],(R0)[R1]   #                       a[i] = a[i-1];
!       MOVW    R2,-2(R0)[R1]         #                       a[i-1] = tmp;
        ACBL    $1,$0,R1,L2           #_______                if (i > 1) i--;
        SUBL2   $2,R1                 #_/  __         } else i++;
  L2:   AOBLSS  8(AP),R1,L1           #___/__ } while (i < n);
```