



Министерство науки и высшего образования Российской Федерации
Мытищинский филиал
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Космический

КАФЕДРА «Прикладная математика, информатика и вычислительная техника» КЗ-МФ

Лабораторная работа №4

ПО ДИСЦИПЛИНЕ:

Организация ЭВМ и систем

Студент КЗ-66Б
Дмитриевич
(Группа)

(Подпись, дата) Чернов Владислав
(И.О.Фамилия)

Студент КЗ-66Б
(Группа)

(Подпись, дата) Братов Аким Романович
(И.О.Фамилия)

Преподаватель

(Подпись, дата) Ефремов Николай Владимирович
(И.О.Фамилия)

2025 г.

Цель работы

Изучить, как символьная информация представляется в памяти процессорной системы, директивы ассемблера для работы с символьной информацией, научиться выводить её на LCD дисплей стенда.

Приложенные материалы

Команды «load», «store»

Команды загрузки/сохранения предназначены для пересылки данных между регистрами общего назначения процессора и оперативной памятью (устройствами ввода/вывода). Они относятся к формату I-типа. Ниже перечислены сами команды и краткие пояснения.

ldw(load word) rB, byte_offset (rA) – загрузка в регистр rB слова из ОП. Адрес операнда в ОП определяется путем сложения содержимого регистра rA и смещения byte_offset: (rA) + смещение.

stw(store word) rB, byte_offset (rA) – сохранение слова из rB в ОП по адресу (rA) + смещение.

ldb (load byte) – загрузка в регистр процессора байта со знаком из ОП.

ldbu (load byte unsigned) – загрузка в регистр процессора байта без знака из ОП.

ldh (load halfword) – загрузка в регистр процессора полуслова со знаком из ОП.

ldhu (load halfword unsigned) – загрузка в регистр процессора полуслова без знака из ОП. При загрузке операнда со знаком в 32-битный регистр операнд дополняется до 32 разрядов знаковым разрядом. При загрузке операнда без знака в 32-битный регистр операнд дополняется до 32 разрядов нулями.

stb (store byte) – сохранение байта из регистра процессора в ОП.

sth (store halfword) – сохранение полуслова из регистра процессора в ОП. Представленные выше две команды выполняют сохранение младшего байта (полуслова) регистра в ОП. Ниже приведены команды загрузки/сохранения операнда в устройствах ввода/вывода.

ldwio (load word I/O) – загрузка слова из устройства ввода/вывода в регистр процессора.

ldbio (load byte I/O) – загрузка байта со знаком из устройства ввода/вывода в регистр процессора.

ldbuio (load byte unsigned I/O) – загрузка байта без знака из устройства ввода/вывода в регистр процессора.

ldhio (load halfword I/O) – загрузка полуслова со знаком из устройства ввода/вывода в регистр процессора.

ldhuio (load halfword I/O) – загрузка полуслова без знака из устройства ввода/вывода в регистр процессора.

stwio (store word I/O) – сохранение слова из регистра процессора в устройстве ввода/вывода.

stbio (store byte I/O) – сохранение байта из регистра процессора в устройстве ввода/вывода.

sthio (store halfword I/O) – сохранение полуслова из регистра процессора в устройстве ввода/вывода.

Команды загрузки/сохранения операнда в устройствах ввода/вывода выполняются без обращения к кэш памяти

Исходный файл программы lab4_part3.s

```
.text /* объявляем сегмент кода */

.equ A_start, 0x8000000 /* начальный адрес фрагмента ОП */
.equ A_end, 0x81FFFFFF /* конечный адрес фрагмента ОП */
.equ step, 0x1 /* шаг для изменения числа - заполнителя */
.equ number, 0x0 /* число - заполнитель */
.equ red, 0x10000000 /* адрес красных светодиодов */
.equ green, 0x10000010 /* адрес зеленых светодиодов */
.equ on, 0x3FFFF /* набор для зажигания светодиодов */

.global _start
_start:

    movia r5, number /* в r5 число для записи в ОП */
    movia r6, A_start /* в r6 начальный адрес фрагмента ОП */
    movia r8, A_end /* в r8 конечный адрес фрагмента ОП */
    movia r3, red /* адрес красных светодиодов в r3 */
    movia r4, green /* адрес зеленых светодиодов в r4 */
    movia r2, on /* набор для зажигания в r2 */

    stwio r0, (r3) /* гасим красные светодиоды */
    stwio r0, (r4) /* гасим зеленые светодиоды */

p:

    stw r5, (r6) /* число из r5 в ОП */
    ldw r7, (r6) /* число из ОП в r7 */
    bne r5, r7, ledr /* если не совпали-зажигаем красные светодиоды */
    addi r5, r5, step /* увеличиваем число-заполнитель на step */
    addi r6, r6, 4 /* адресуем следующее слово в ОП */
    bgt r6, r8, ledg /* если проверили весь диапазон адресов, зажигаем зеленые светодиоды */
    br p /* переходим на следующее обращение к ОП */

ledr: /* зажигаем красные светодиоды */
    stwio r2, (r3)
    br stop

ledg: /* зажигаем зеленые светодиоды */
    stwio r2, (r4)
```

```
stop: br stop /* завершаем программу */
```

```
.end
```

Выполнение заданий лабораторной работы

1. Фрагмент программы, которая будет сохранять слово в ОП по начальному адресу статической памяти, используемой в процессорной системе. Затем выполняется сохранение этого же слова по смещенному на единицу адресу. Затем по адресу, смещенному на 2, на 3, на 4 и на 5, соответственно. Код и состояние памяти отображены на фото.

```
.text
.global _start
_start:
    movia r4, 0x8000000
    movia r5, 0x12345678

    stw r5, (r4)      # Сохраняем в ОП
    stw r5, 1(r4)     # Сохраняем в ОП
    stw r5, 2(r4)     # Сохраняем в ОП
    stw r5, 3(r4)     # Сохраняем в ОП
    stw r5, 4(r4)     # Сохраняем в ОП
    stw r5, 5(r4)     # Сохраняем в ОП

STOP:
    br STOP

.end
```

Address (hex)	+0x0	+0x4	+0x8	+0xc
0x00800000	12345678	12345678	5d550fd5	5826fb13
0x00800010	1fa2099a	64eace23	4744e3c6	33d24dc8
0x00800020	0b7407ae	1b61ab8c	26f03b32	8adc14d2
0x00800030	57900faa	17a02e42	be7a0eba	2224a650
0x00800040	0ed04680	07f0156e	1fc01f90	80821bcd
0x00800050	27eb2d2b	8dbadda2	0c1088f5	bfb247f4
0x00800060	c9ff6508	97f2825f	04508dd1	22a10e9c
0x00800070	36603bd9	6d727fe8	8c145c66	fe7c8bdc
0x00800080	0200828a	057a0dd1	0ab96d54	b2102b80
0x00800090	995f05b8	4ee8cd26	efe9a03b	4f701b6a
0x008000a0	34a00612	8dc967a0	8bd557ea	1f101182
0x008000b0	3da2163a	05c2ae38	5365ab54	0da3faf4
0x008000c0	ce723714	ff760519	2258a5e5	c6a2a9d1
0x008000d0	35792784	3aff2732	7ef50a30	73942776
0x008000e0	45b10799	4f221d84	18d30f94	27f00ffc
0x008000f0	af261702	5dd10a40	ebe82cba	8a320f98

2. Наблюдаем за поведением программы. Порядок байт — little endian. Поэтому строка перевёрнута. Здесь мы используем «word» (4 байта), поэтому текст отображается полностью на сдвиге +4. При этом стоит заметить, что отображение «наполовину» невозможно. Текст всегда будет отображаться с выравниванием, соответствующему типу данных.

+0x0	+0x4	+0x8	+0xc
00 00 00 00	78 56 34 12	00 00 00 00	00
00 00 00 00	00 00 00 00	00 00 00 00	00
00 00 00 00	00 00 00 00	00 00 00 00	00
00 00 00 00	00 00 00 00	00 00 00 00	00
00 00 00 00	00 00 00 00	00 00 00 00	00
00 95 03 6e	b1 f9 05 ba	bb d5 5f ce	a7

3. Переписываем программу, используя уже «half-word» (2 байта). Теперь отображается только половина текста.

```
.text
.global _start
_start:
    movia r4, 0x8000000
    movia r5, 0x23456789

    stw r5, (r4)      # Сохраняем в ОП
    stw r5, 4(r4)     # Сохраняем в ОП
    ldh r10, 0(r4)
    ldh r11, 1(r4)
    ldh r12, 2(r4)
    ldh r13, 3(r4)
    ldh r14, 4(r4)
    ldh r15, 5(r4)
    ldh r17, 8(r4)

STOP:
    br STOP

.end
```

+0x0	+0x4	+0x8	+0xc
0x00800000	0000 0000	5678 0000	0000 0000
0x00800010	0000 0000	0000 0000	0000 0000
0x00800020	0000 0000	0000 0000	0000 0000
0x00800030	0000 0000	0000 0000	0000 0000
0x00800040	0000 0000	0000 0000	0000 0000

4. Перепишем код ещё раз, но в этот раз для размера в 1 байт.

```
.text
.global _start
_start:
    movia r4, 0x80000000
    movia r5, 0x12345678

    stb r5, (r4)      # Сохраняем в ОП
    stb r5, 1(r4)     # Сохраняем в ОП
    stb r5, 2(r4)     # Сохраняем в ОП
    stb r5, 3(r4)     # Сохраняем в ОП
    stb r5, 4(r4)     # Сохраняем в ОП
    stb r5, 5(r4)     # Сохраняем в ОП
    stb r5, 8(r4)     # Сохраняем в ОП

STOP:
    br      STOP

.end
```

+0x0	+0x4	+0x8	+0xc
00 00 78 78	78 78 00 00	78 00 00 00	00 00 00 00
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

5. Модифицируйте программу из части 1 таким образом, чтобы в программе теперь выполнялось чтение из последовательных адресов ОП слов, полуслов, байтов аналогично тому, как это делалось в пунктах 1-3 первой части. Перед выполнением команд чтения из ОП, для наглядности, рекомендуется предварительно выполнить запись в последовательные ячейки памяти следующих двух слов 0x23456789, 0xabcdef01.

```
.text
.global _start
_start:
    movia r4, 0x80000000
    movia r5, 0x23456789

    stw r5, (r4)      # Сохраняем в ОП
    movia r5, 0xabcdef01
    stw r5, 4(r4)     # Сохраняем в ОП

    ldh r10, 0(r4)
    ldh r11, 1(r4)
    ldh r12, 2(r4)
    ldh r13, 3(r4)
    ldh r14, 4(r4)
    ldh r15, 5(r4)
    ldh r17, 8(r4)

STOP:
    br      STOP

.end
```

	+0x0	+0x4	+0x8
0x00000000	89 67 45 23	01 ef cd ab	00 00 00 00
0x00000010	00 00 00 00	00 00 00 00	00 00 00 00
0x00000020	00 00 00 00	00 00 00 00	00 00 00 00
0x00000030	00 00 00 00	00 00 00 00	00 00 00 00
0x00000040	00 00 00 00	00 00 00 00	00 00 00 00

Выполните программу, используя в качестве ОП статическую память. С помощью вкладки Memory приложения ARM убедитесь, что заполнение ОП произошло в соответствии с заданным вариантом, причем во всем заданном диапазоне. Выполните снимки экрана монитора ПК, отражающие содержимое ОП в начале и в конце диапазона и поместите их в отчет.

```
.text
/* объявляем сегмент кода */
.equ A_start, 0x000468 /* начальный адрес фрагмента ОП */
.equ A_end, 0x3FFFFFFF /* конечный адрес фрагмента ОП */
.equ step, 0x2 /* шаг для изменения числа - заполнителя */
.equ number, 0x9 /* число - заполнитель */
.equ red, 0x10000000 /* адрес красных светодиодов */
.equ green, 0x100000010 /* адрес зеленых светодиодов */
.equ on, 0x3FFFF /* набор для зажигания светодиодов */

.global _start
_start:
    movia r5, number /* в r5 число для записи в ОП */
    movia r6, A_start /* в r6 начальный адрес фрагмента ОП */
    movia r8, A_end /* в r8 конечный адрес фрагмента ОП */

    movia r3, red /* адрес красных светодиодов в r3 */
    movia r4, green /* адрес зеленых светодиодов в r4 */
    movia r2, on /* набор для зажигания в r2 */

    stwio r0, (r3) /* гасим красные светодиоды */
    stwio r0, (r4) /* гасим зеленые светодиоды */

p:
    andi r5, r5, 0x000000ff /* число из r5 в ОП */
    ldbu r7, (r6) /* число из ОП в r7 */

    bne r5, r7, ledr /* если не совпали - зажигаем красные светодиоды */

    addi r5, r5, step /* увеличиваем число-заполнитель на step */
    addi r6, r6, 1 /* адресуем следующее слово в ОП */

    bgt r6, r8, ledg /* если проверили весь диапазон адресов, зажигаем зеленые светодиоды */
    br p /* переходим на следующее обращение к ОП */

ledr: /* зажигаем красные светодиоды */
```

Внесите изменения в программу таким образом, чтобы заполнение ОП выполнялось полусловами. Добейтесь правильного завершения программы, сопровождаемое загоранием зеленых светодиодов. Включите в отчет снимки экрана ПК, подтверждающие правильное заполнение ОП и листинг отлаженной программы.

```
.global _start
_start:
    movia r5, number    /* в r5 число для записи в ОП */
    movia r6, A_start    /* в r6 начальный адрес фрагмента ОП */
    movia r8, A_end      /* в r8 конечный адрес фрагмента ОП */

    movia r3, red        /* адрес красных светодиодов в r3 */
    movia r4, green      /* адрес зеленых светодиодов в r4 */
    movia r2, on         /* набор для зажигания в r2 */

    stwio r0, (r3)       /* гасим красные светодиоды */
    stwio r0, (r4)       /* гасим зеленые светодиоды */

p:
    andi r5, r5, 0x000000ff /* число из r5 в ОП */
    stb r5, (r6)          /* число из ОП в r7 */
    ldbu r7, (r6)         /* число из ОП в r7 */

    bne r5, r7, ledr /*если не совпали-зажигаем красные светодиоды*/

    addi r5, r5, step     /* увеличиваем число-заполнитель на step */
    addi r6, r6, 1        /* адресуем следующее слово в ОП */

    bgt r6, r8, ledg      /* если проверили весь диапазон адресов,
                           зажигаем зеленые светодиоды */
    br p                 /* переходим на следующее обращение к ОП */

ledr:
    stwio r2, (r3)        /* зажигаем красные светодиоды */
    br stop
ledg:
    stwio r2, (r4)        /* зажигаем зеленые светодиоды */

stop:
    br stop              /* завершаем программу */
.end
```

Внесите изменения в программу таким образом, чтобы заполнение ОП выполнялось байтами. Добейтесь правильного завершения программы, сопровождаемое загоранием зеленых светодиодов. Включите в отчет снимки экрана ПК, подтверждающие правильное заполнение ОП и листинг отлаженной программы.

```
.global _start
_start:
    movia r5, number    /* в r5 число для записи в ОП */
    movia r6, A_start    /* в r6 начальный адрес фрагмента ОП */
    movia r8, A_end      /* в r8 конечный адрес фрагмента ОП */

    movia r3, red        /* адрес красных светодиодов в r3 */
    movia r4, green      /* адрес зеленых светодиодов в r4 */
    movia r2, on         /* набор для зажигания в r2 */

    stwio r0, (r3)       /* гасим красные светодиоды */
    stwio r0, (r4)       /* гасим зеленые светодиоды */

p:
    andi r5, r5, 0x000000ff /* число из r5 в ОП */
    stb r5, (r6)          /* число из ОП в r7 */
    ldbu r7, (r6)         /* число из ОП в r7 */

    bne r5, r7, ledr /*если не совпали-зажигаем красные светодиоды*/

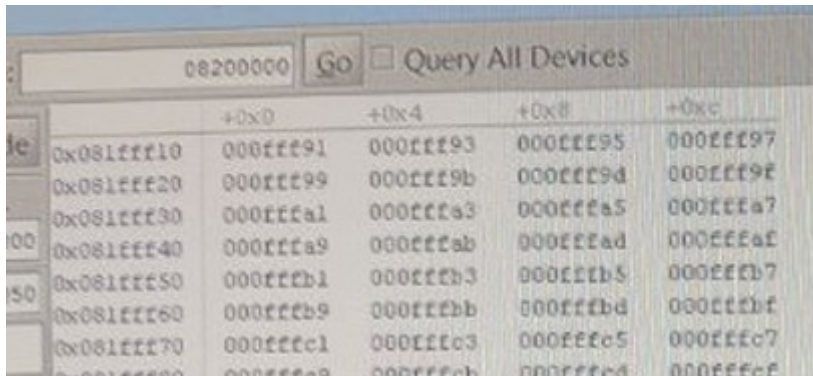
    addi r5, r5, step     /* увеличиваем число-заполнитель на step */
    addi r6, r6, 1        /* адресуем следующее слово в ОП */

    bgt r6, r8, ledg      /* если проверили весь диапазон адресов,
                           зажигаем зеленые светодиоды */
    br p                 /* переходим на следующее обращение к ОП */

ledr:
    stwio r2, (r3)        /* зажигаем красные светодиоды */
    br stop
ledg:
    stwio r2, (r4)        /* зажигаем зеленые светодиоды */

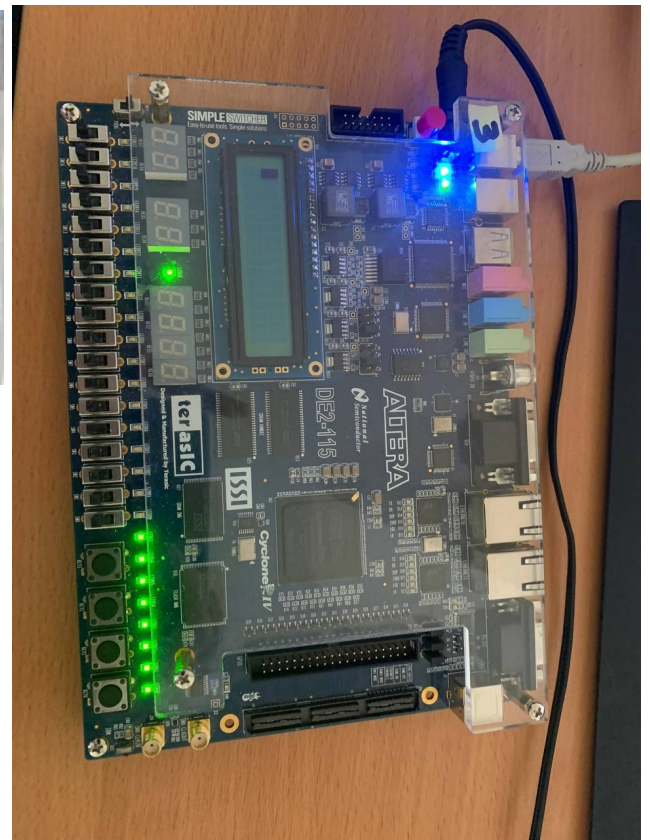
stop:
    br stop              /* завершаем программу */
.end
```


Проверьте правильность работы программы, используя в качестве ОП динамическую память. Учтите, что в динамической памяти размещена сама программа. Поэтому в качестве начального адреса ОП используйте адрес ячейки, следующей сразу за программой. Обратите внимание на существенное увеличение времени выполнения программы. Включите в отчет свои соображения по этому поводу



The screenshot shows a debugger window with a search bar at the top containing '08200000' and a 'Go' button. Below the search bar is a table of memory addresses and their corresponding values. The table has four columns: '+0x0', '+0x4', '+0x8', and '+0xc'. The rows show addresses from 0x081fff10 to 0x081fff80. The values are hexadecimal numbers.

	+0x0	+0x4	+0x8	+0xc
0x081fff10	000ffff91	000ffff93	000ffff95	000ffff97
0x081fff20	000ffff99	000ffff9b	000ffff9d	000ffff9f
0x081fff30	000ffffa1	000ffffa3	000ffffa5	000ffffa7
0x081fff40	000ffffa9	000ffffab	000ffffad	000ffffaf
0x081fff50	000ffffb1	000ffffb3	000ffffb5	000ffffb7
0x081fff60	000ffffb9	000ffffbb	000ffffbd	000ffffbf
0x081fff70	000ffffc1	000ffffc3	000ffffc5	000ffffc7
0x081fff80	000ffffc9	000ffffcb	000ffffcd	000ffffcf



Заключение

В ходе выполнения лабораторной работы №4, студенты обрели следующие навыки:

1. Создавать и использовать программные средства для тестирования компонентов ОП процессорной системы.
2. Понимание, каким образом выполняется взаимодействие процессора с ОП и с устройствами ввода вывода и принципа работы команд load/store