Зертханалық жұмыс №1

Пән: Компьютер архитектурасы және операциялар келісімділігі

Тақырыбы: Төменгі деңгейдегі ассемблер Mars MIPS симуляторында кіріспе есептер

Мақсаты:

- Mars MIPS симуляторының жұмыс принципін меңгеру;
- Ассемблер тілінде алғашқы бағдарламаларды жазу;
- Регистрлердің жұмысын практикада көру;
- Жадымен жұмыс жасауды үйрену.

Қысқаша теория

Компьютер архитектурасында бағдарламалау тек жоғары деңгейлі тілдерде ғана емес, сонымен қатар төменгі деңгейлі ассемблер тілінде де орындалады. Ассемблер – процессордың машиналық командаларына барынша жақын жазылатын бағдарлама тілі. Бұл тілде әрбір команда нақты бір процессорлық нұсқауды орындайды. Ассемблерді үйренудің басты артықшылығы – студент компьютердің қалай жұмыс істейтінін терең түсінеді, яғни регистрлердің, жадының, арифметикалық және логикалық блоктардың байланысын көреді.

MIPS архитектурасы – RISC (Reduced Instruction Set Computer) принципіне негізделген процессорлардың нұсқаулар жүйесі. Оның басты ерекшелігі – командалардың қарапайым әрі біркелкі құрылымда болуы. Әрбір команда бір машиналық такт ішінде орындалады. Бұл архитектура білім беру саласында жиі қолданылады, себебі студенттер үшін түсінікті әрі тиімді.

MIPS процессорының негізгі жұмыс элементтері – **регистрлер**. Олар процессордың ішінде орналасқан өте жылдам жады ұяшықтары. Регистрлер бірнеше топқа бөлінеді:

- $t0–$t9 уақытша регистрлер – есептеулер кезінде қолданылады;
- $s0–$s7 сақталатын регистрлер – функциялар арасында деректерді сақтайды;
- $a0–$a3 аргумент регистрлері – функцияларға параметр беруге арналған;
- $v0–$v1 қайтару регистрлері – операцияның немесе функцияның нәтижесін сақтайды.

Бағдарлама жазу барысында .data және .text деген екі негізгі бөлім қолданылады. .data бөлімінде айнымалылар мен жолдық хабарламалар анықталады. Мысалы:

```
.data
msg: .asciiz "Hello, MIPS!"
```

Бұл жол "Hello, MIPS!" мәтінін жадта сақтайды және оған msg деген белгі (label) беріледі.

.text бөлімінде бағдарламаның командалары жазылады. Бағдарламаның басталу нүктесі ретінде main: белгісі жиі қолданылады. Барлық негізгі командалар осы бөлімде орналасады.

Syscall – MARS MIPS симуляторындағы арнайы жүйелік шақыру механизмі. Ол әртүрлі енгізу-шығару қызметтерін орындауға мүмкіндік береді. Мысалы, li $v0, 1 – санды экранға шығару қызметі, ал li $v0, 4 – жолды шығару қызметі. li $v0, 5 арқылы сан енгізуге болады. Әр syscall шақырылғанда, қандай операция орындалатыны $v0 регистріндегі мәнге байланысты болады.

Арифметикалық командалар:

- add $t0, $t1, $t2 – $t0 = $t1 + $t2
- sub $t0, $t1, $t2 – $t0 = $t1 – $t2
- mul $t0, $t1, $t2 – $t0 = $t1 * $t2
- div $t0, $t1, $t2 – $t0 = $t1 / $t2

Жадымен жұмыс істеу командалары:

- lw $t0, var – жадтан дерек оқу (load word)
- sw $t0, var – деректі жадқа жазу (store word)

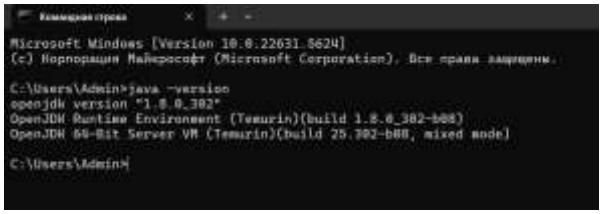
MARS MIPS симуляторында бағдарламаны орындау кезінде студент тек қана нәтижені емес, сонымен қатар регистрлер мен жадыда қандай өзгерістер болғанын көре алады. Бұл төменгі деңгейлі бағдарламалаудың басты артықшылығы болып табылады, өйткені студент бағдарламаның қалай орындалатынын нақты қадамдар бойынша бақылай алады.

Осы зертханалық жұмыста қарапайым амалдар – мәтін шығару, қосу, азайту және жадымен жұмыс істеу қарастырылады. Бұл кіріспе есептер кейінгі күрделі тапсырмаларға негіз болады.

MARS симуляторын ашу үшін Java Runtime Environment пакеті қажет болады, алдымен осы программаны https://www.java.com/ru/download/ сілтемесі бойынша жазып алып, MARS файлын төмендегі сілтеме сайтынан алып

https://dpetersanderson.github.io/download.html копьютер жұмыс столынан Java көмегімен ашыңыз!!!

Java сіздің компьютеріңізде орнатылғанын білу үшін осы команданы жазыңыз!



Мысалдар

Мысал 1. Жолды экранға шығару

Жаңа ассемблер файлын жасау

1. MARS ашылғаннан кейін: **File → New**
2. .asm файлында код жазыңыз, мысалы:

```
.data
msg: .asciiz "Hello, MIPS!"   # Шығатын мәтінді анықтау

.text
main:
   li $v0, 4       # 4 қызметі – жолды экранға шығару
   la $a0, msg     # $a0 регистріне msg жолының адресін жүктеу
   syscall         # жүйелік шақыру (мәтінді шығарады)
```

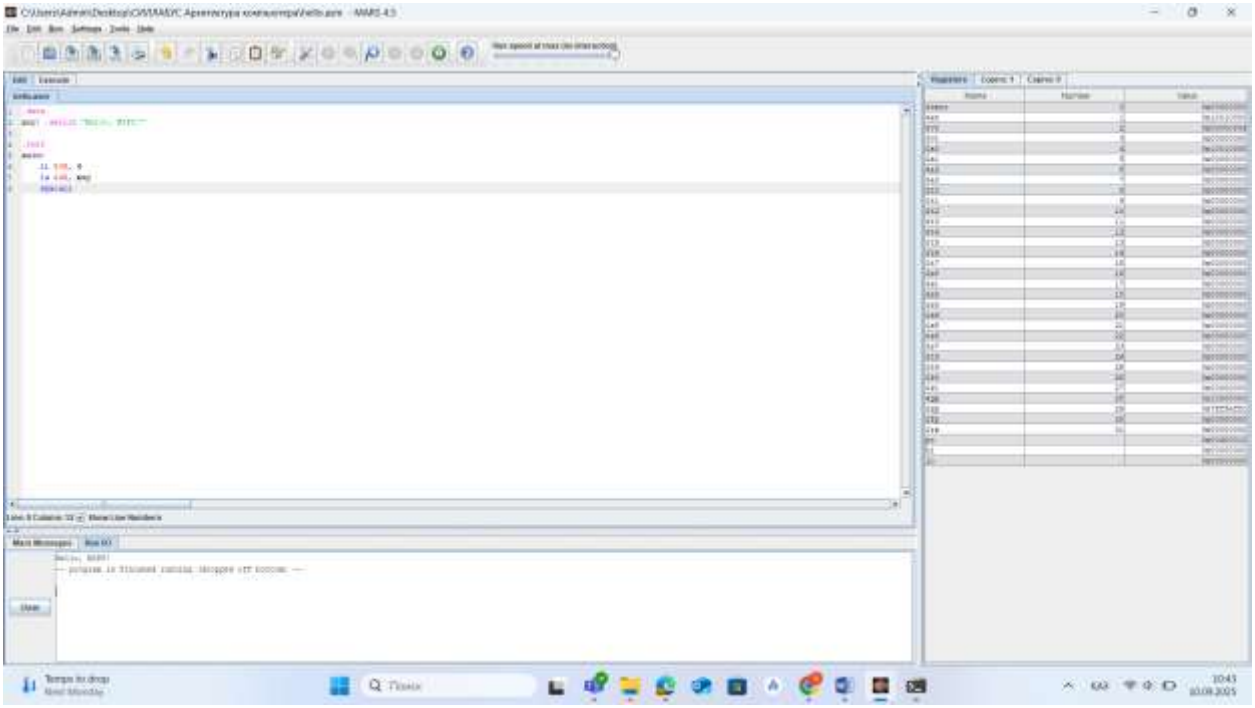
3. **File → Save As → hello.asm** (файл атын беріңіз)

5. Кодты іске қосу

1. **Assemble → Assemble All** (немесе F3) – код жинақталады.
2. **Run → Go** (немесе F5) – бағдарлама орындалады.
3. Терезеде (Console) нәтиже шығады:

Hello, MIPS!

Түсіндірме: Бұл бағдарлама экранға "Hello, MIPS!" деген мәтінді шығарады.



Мысал 2. Қосу операциясы

```
.text
main:
   li $t0, 7          # Бірінші сан
   li $t1, 5          # Екінші сан
   add $t2, $t0, $t1   # t2 = t0 + t1 → қосындыны есептеу

   move $a0, $t2       # нәтижені a0-ге көшіру
   li $v0, 1           # 1 қызметі – бүтін санды шығару
   syscall             # нәтижені экранға шығару
```

Түсіндірме: Екі сан қосылып, нәтиже экранға шығады.

№1 зертханалық жұмыстың тапсырмалары.

1. Экранға өз атыңызды және топ нөміріңізді шығарыңыз.
2. Екі санды қосып, нәтижесін экранға шығарыңыз.
3. Екі санды азайтып, нәтижесін экранға шығарыңыз.
4. .data бөлімінде бүтін сан сақтап, оған 10 қосып, нәтижесін экранға шығарыңыз.
5. Жадыда екі санды сақтап, оларды қосып, нәтижесін экранға шығаратын бағдарлама құрыңыз.