Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ

Звіт

до лабораторної роботи № 6

з дисципліни «Комп'ютерні системи»
на тему: «Побудова функціональної схеми процесорів архітектури RISC CPU»

Варіант №1

Виконав:

ст.гр. КІ-38

Александрова Р.-Є.О.

Прийняв:

Старший викладач

Козак Н. Б.

Львів 2022

Мета роботи: Навчитись розробляти нові функціональні модулі з метою розширення функціонування базової архітектури процесорів RISC, що відповідає програмній моделі, а також створення функціональної схеми

Завдання:

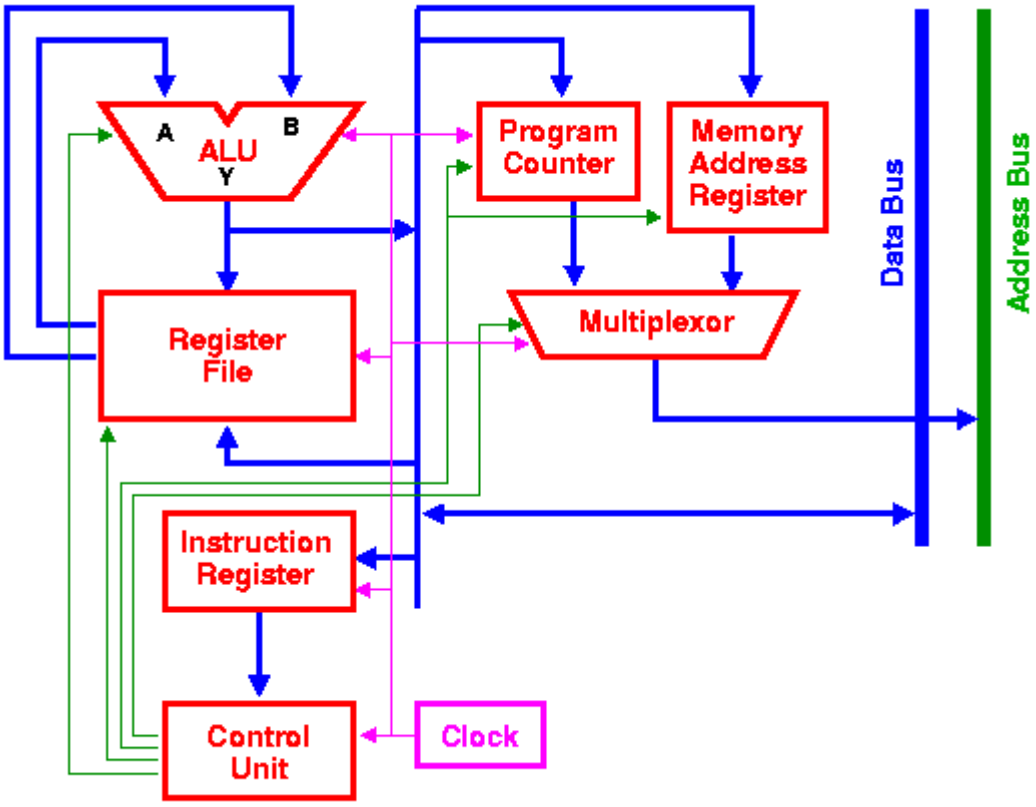
Згідно варіанту в журналі та узгоджено з викладачем обрати один з процесорів заданого сімейства на базі RISC архітектури, в звіті навести основну інформацію по процесору, схему та характеристику.

Внести необхідні зміни в проект з 5-ї лабораторної (типи кешів, розмірності шин даних, назви та типи сигналів, операції, буферні модулі і т.д.) з метою реалізації фукнціоналу з обраного процесору. В звіті представити коротко внесені зміни з описом ходу роботи, а також кінцеву функціональну схему програмної моделі.

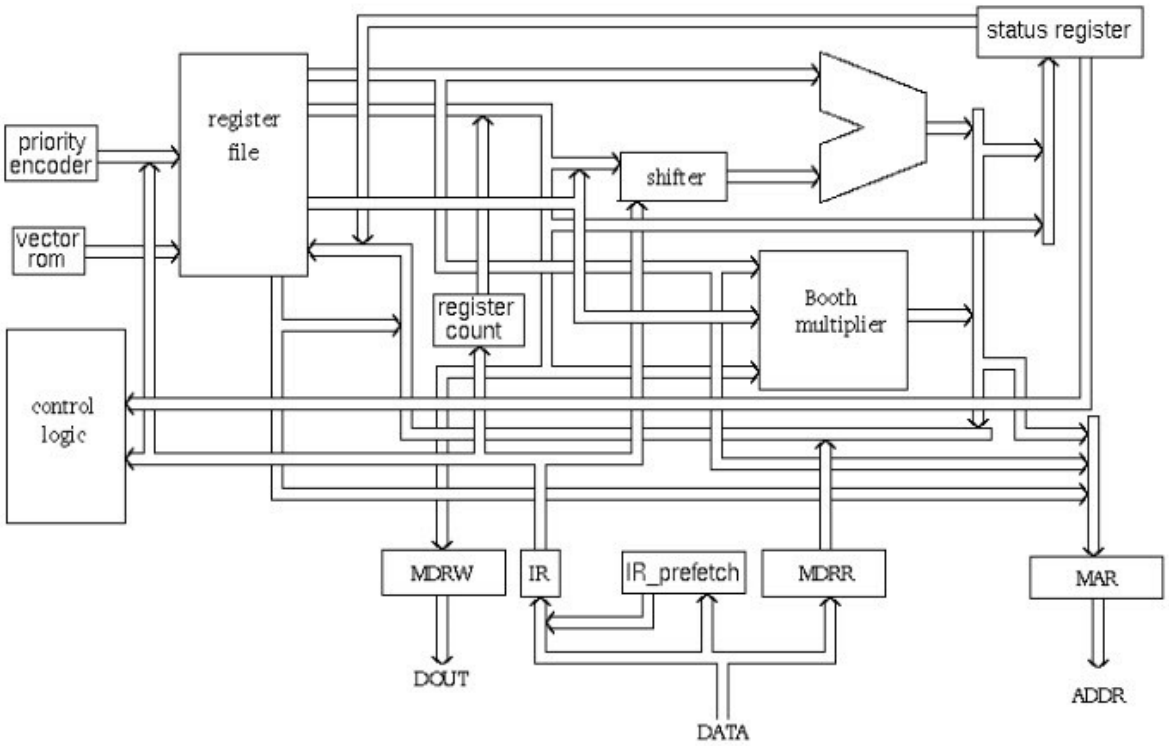
1) ARM сімейство процесорів

Хід роботи:

Функціональна схема RISC процесора:



Функціональна схема процесора ARM:



Характеристики ARM процесора:

- Багатопроцесорні системи -

Процесори ARM розроблені таким чином, щоб їх можна було використовувати у випадках багатопроцесорних систем, де для обробки інформації використовується більше одного процесора. Перший процесор AMP, представлений під назвою ARMv6K, мав можливість підтримувати 4 ЦП разом зі своїм апаратним забезпеченням.

- Тісно пов'язана пам'ять -

Пам'ять процесорів ARM тісно пов'язана. Це має дуже швидкий час реакції. Він має низьку затримку (швидку відповідь), що також можна використовувати у випадках непередбачуваної роботи кеш-пам’яті.

- Управління пам'яттю -

Процесор ARM має розділ керування. Це включає в себе блок управління пам'яттю і блок захисту пам'яті. Ці системи управління стають дуже важливими для ефективного управління пам'яттю.

- Технологія Thumb-2 –

Технологія Thumb-2 була представлена в 2003 році і використовувалася для створення набору інструкцій змінної довжини. Вона розширює 16-розрядні

інструкції початкової технології Thumb до 32-розрядних інструкцій. Має кращу продуктивність, ніж раніше використовувана технологія Thumb.

- Час виконання одного циклу -

Процесор ARM оптимізовано для кожної інструкції на CPU. Кожна інструкція має фіксовану довжину, що дає час для отримання майбутніх інструкцій перед виконанням поточної інструкції.

- Конвеєризація -

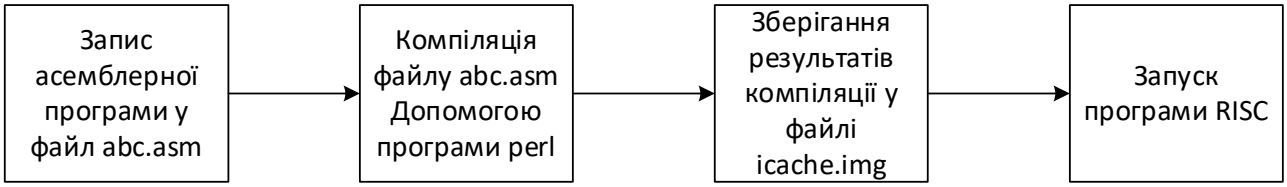
Обробка інструкцій здійснюється паралельно за допомогою конвеєрів. Інструкції розбиваються та декодуються на одному етапі конвеєра. Конвеєр просувається один крок за раз, щоб збільшити пропускну здатність (швидкість обробки).

- Велика кількість регістрів –

Велика кількість регістрів використовується в процесорі ARM для запобігання великої кількості взаємодій з пам'яттю. Реєстри містять дані та адреси. Вони діють як локальне сховище для всіх операцій.

Переваги	Недоліки
Низька ціна	Несумісний із програмами x86, такими як ОС Windows.
Простий дизайн	У деяких процесорах швидкість обмежена
Низькі вимоги до потужності	Простіший набір інструкцій може бути недостатнім для більш важких робочих навантажень
Кращий термін служби батареї	ARM має обмежену обчислювальну здатність
Низька теплогенерація	Продуктивність залежить від здатності програміста правильно виконувати і часто вимагає високої кваліфікації програмістів.

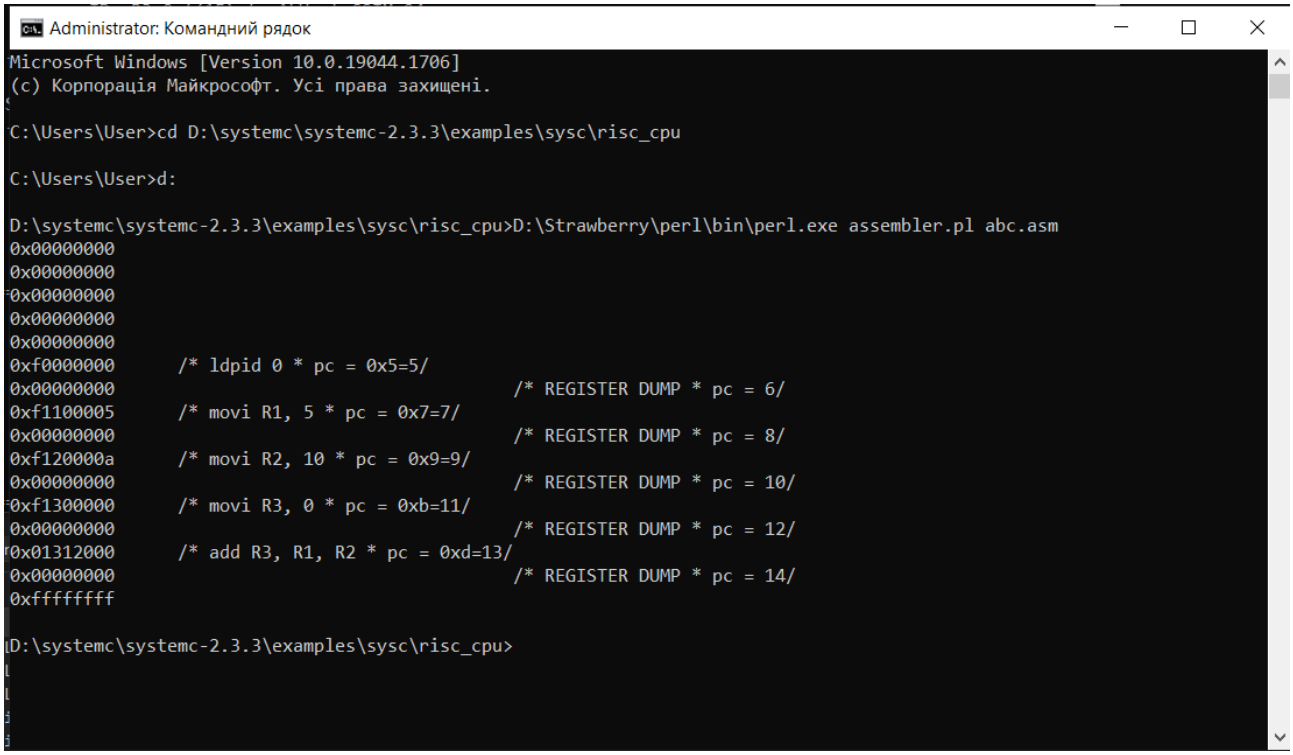
Схема виконання програми:



Вміст файлу abc.asm:

```
ldpid 0
movi  R1, 5
movi  R2, 10
movi  R3, 0
add   R3, R1, R2
```

Компіляція файлу abc.asm за допомогою perl.exe:



Вміст файлу icache.img:

icache.img

```
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0xf0000000
0x00000000
0xf1100005
0x00000000
0xf120000a
0x00000000
0xf1300000
0x00000000
0x01312000
0x00000000
0xffffffff
```

Результат роботи програми:

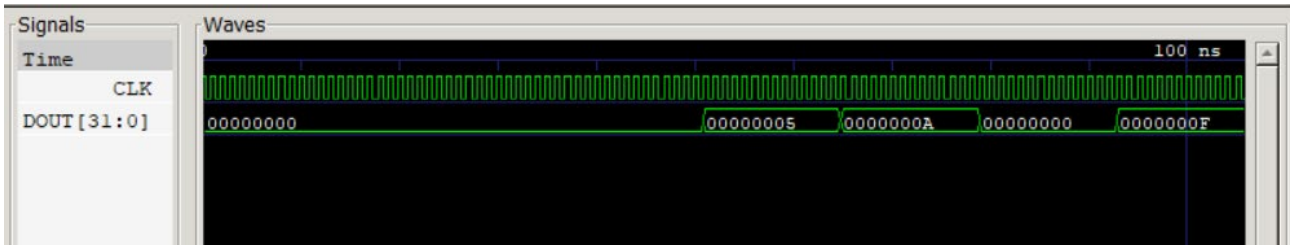
Microsoft Visual Studio Debug Console

```
========================================================================


------------------------
IFU : mem=0x1312000
IFU : pc= d at CSIM 89 ns
------------------------
                      ------------------------------
                      ID: R3= R1(=5)+R2(=10)
                        : at CSIM 91 ns
                      ------------------------------
                                                    ------------------------------
                                                    ALU : op= 3 A= 5 B= 10
                                                    ALU : R= 15-> R3 at CSIM 93 ns
                                                    ------------------------------
                      ------------------------------
                      ID: R3=0xf(15) fr ALU at CSIM 94 ns
                      ------------------------------
------------------------
IFU : mem=0x0
IFU : pc= e at CSIM 96 ns
------------------------
                      ------------------------------


                      ******************************
                      ID: REGISTERS DUMP at CSIM 98 ns
                      ******************************
REG :==================================================================
  R 0(00000000)    R 1(00000005)    R 2(0000000a)    R 3(0000000f)
  R 4(00000004)    R 5(00000005)    R 6(00000006)    R 7(fcf0fdef)
  R 8(00000008)    R 9(00000009)    R10(00000010)    R11(0000ff31)
  R12(0000ff12)    R13(00000013)    R14(00000014)    R15(00000015)
  R16(00000016)    R17(00fe0117)    R18(00fe0118)    R19(00fe0119)
  R20(00fe0220)    R21(00fe0321)    R22(00fe0322)    R23(00ff0423)
  R24(00ff0524)    R25(00ff0625)    R26(00ff0726)    R27(00ff0727)
  R28(00f70728)    R29(00000029)    R30(00000030)    R31(00000031)
========================================================================
```

Часова діаграма:



Висновок: Виконуючи дану лабораторну роботу я вивчив структурну схему об`єкта (RISC CPU), що відповідає програмній моделі, і побудував функціональну схему. Також виконав додаткове завдання для того щоб перевірити правильність виконання програми.