FRI-2G.303-1-CCT1-06

A DIDACTIC PIPELINED PROCESSOR [6]

Principal Assistant Nikolay Kostadinov, PhD
Department of Computing,
"Angel Kanchev" University of Ruse
E-mail: nkostadinov@uni-ruse.bg

Abstract: *The inherent complexity of modern processors presents a number of challenges in the learning and teaching process for both computer engineering students and instructors of computer architecture courses. To facilitate students' understanding of the architectural aspects of computer systems, two complementary approaches are typically combined: the study of real-world processors and the study of simplified processor models. This paper presents the design of a didactic RISC processor that employs a 4-stage instruction execution pipeline. With a focus on educational usability, the instruction set and addressing modes have been deliberately simplified to support clarity and ease of understanding. The processor's pipelined execution model is described, and the functionality of each pipeline stage is outlined. Additionally, it discusses the techniques used to mitigate data hazards and minimize pipeline stalls, ensuring more efficient instruction flow during execution. The specification, modeling, simulation, and implementation of the processor are carried out using the integrated design environment Xilinx Vivado Design Suite, and the testing is performed on the Basys 3 development board.*

Keywords: *Computer Architecture, Didactic CPU, FPGA.*

ВЪВЕДЕНИЕ

Присъщата сложност на съвременните процесори поставя редица предизвикателства в обучението, както пред студентите по компютърното инженерство, така и пред преподавателите по дисциплини, свързани с компютърните архитектури. С цел осмисляне от страна на студентите на архитектурните аспекти на компютърните системи в обучението най-често се комбинират два подхода: изучаване на конкретни процесори и изучаване на опростени процесорни модели. При първия подход възможностите за изследване на някои разпространени процесорни фамилии (например IA-32/Intel64) са ограничени поради капсулирането на вътрешните им архитектурни механизми, което ги прави 'скрити' за програмиста. При втория подход се акцентира върху базовата функционалност на процесора, като моделът е придружен с изходен код на език за описание на хардуер (най-често VHDL или Verilog). В този случай студентите могат да експериментират с модификации на модела, както и да реализират и тестват съответното процесорно ядро чрез FPGA схеми.

В литературата са описани множество учебни процесорни модели, както на реални, така и на хипотетични процесори. Така например, модификация на популярния процесор MIPS (Patterson, D., & Hennessy, J., 2013), при която 5-стъпалният конвейер е заменен с 3-стъпален, е представена в (Husainali S., Hitesh N., & Abhishek A., 2016). Учебна 16-битова версия на друга популярна архитектура - RISC-V, е разгледана в (Assumpção, J., Ando, O., de Araújo, H., & Gazziro, M. 2024). Изцяло съвместим RISC-V 32-битов процесор, адаптиран за учебни цели, е представен в (Navarro-Torrero, P., Martínez-Rodríguez, M. C., Barriga-Barros, Á., & Brox, P., 2024). В (Chen, Lin & Ma, Xiao & Ji, Xiang, 2023) е описан процеса на проектиране на хипотетичен RISC (Reduced Instruction Set Computer) процесор с наименование LoongArch, използващ 5-стъпален изчислителен конвейер.

В доклада се разглежда проектирането на дидактичен RISC процесор с условно наименование LCP-R16P, използващ 4-стъпален конвейер за изпълнение на инструкциите. С оглед използването на процесора за учебни цели системата инструкции и методите за адресация са максимално опростени. Представен е поточния модел на процесора и са

[6] The paper was presented on 24 October 2025 in section "Communication and Computer Technologies" with original title in Bulgarian: ДИДАКТИЧЕН КОНВЕЙЕРЕН ПРОЦЕСОР

описани отделните стъпала на конвейера. Коментирани са подходите, използвани за избягване на зависимостите, водещи до непроизводителни заемания на конвейера.

ИЗЛОЖЕНИЕ

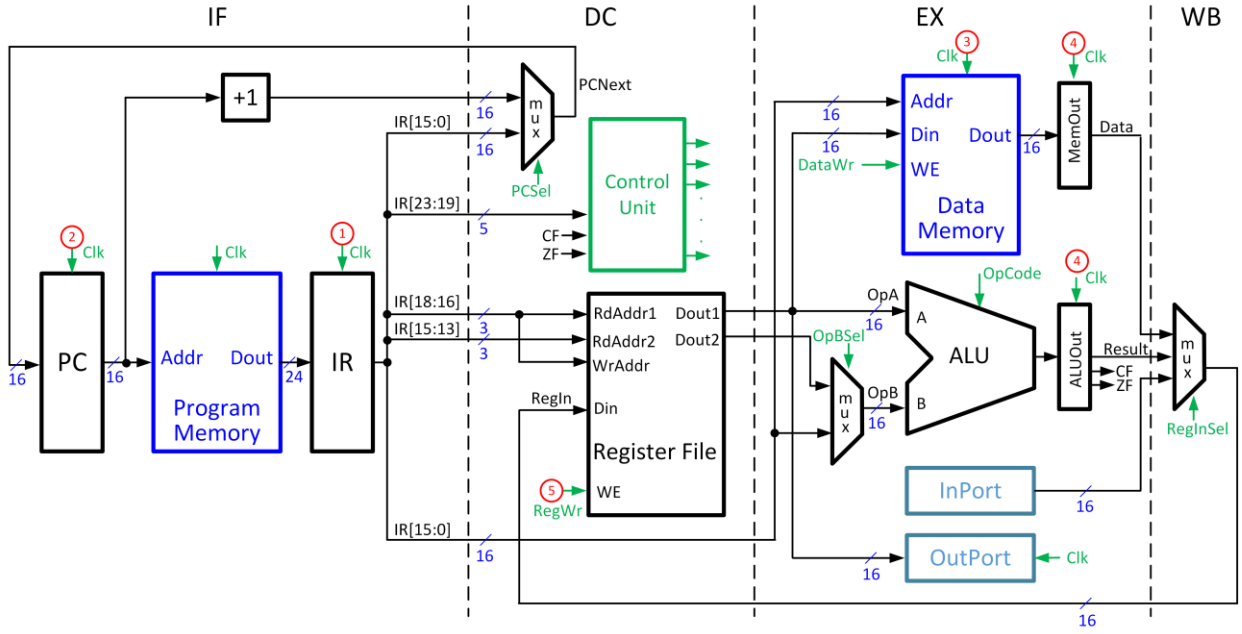
Предложеният процесор е с Харвардска архитектура, характеризираща се с наличие на две отделни адресни пространства - за програми (Program Memory) и за данни (Data Memory). Вътрешната му регистрова памет (Register File) включва осем 16-битови регистъра с общо предназначение. За въвеждане и извеждане на данни са предвидени два 16-разрядни порта: входен (InPort) и изходен (OutPort).

В опростената система инструкции на процесора са включени 16 инструкции, разделени в следните групи:

- Обмен на данни: LOAD, STORE, MOV;
- Аритметична и логическа обработка: ADD, SUB, AND, OR, XOR, SLL, SRL;
- Инструкции за преход: JZ, JNZ, JC, JNC, JMP;
- Празна операция: NOP.

Базовата машина на процесора е от типа „регистър-регистър” (reg-reg, load-store), при която операндите в инструкциите за обработка са задължително в регистри, а достъпът до паметта се извършва само с инструкции LOAD и STORE. Предвидени са три режима на адресация: непосредствена, абсолютна и регистрова. Всички инструкции са 24-битови. Регистърът на състоянието включва 2 бита - флаг за пренос (CF) и флаг за нулев резултат (ZF). Входно/изходните портове InPort и OutPort заемат последните два адреса от паметта, съответно FFFEh и FFFFh.

Поточният модел на многоцикловата реализация на процесора е показан на Фиг. 1.



Фиг. 1. Поточен модел на процесора (многоцикловa реализация)

Този модел цели осмисляне на базовите архитектурни аспекти, като основни функционални устройства на процесора, система инструкции и режими на адресация, взаимодействие на съставните блокове и потоци данни в отделните фази на изпълнение на инструкциите, и служи за основа при преобразуването на многоцикловата реализация на процесора в конвейерна. С оглед на последващо интегриране на изчислителен конвейер са заложени някои ключови характерстики, като:

- Еднаква дължина на всички инструкции;

- Унифициран формат на инструкциите с еднакви полета за отделните видове адресации;
- Разбиване на изпълнението на инструкциите на фази, идентични за всички инструкции: извличане на инструкцията (IF, Instruction Fetch), декодиране на инструкцията (DC, Decode), изпълнение на инструкцията (EX, Execute), обратен запис на резултата в регистровия файл (WB, Write Back);
- Използване на отделни памети за програми и данни, както и самостоятелни функционални устройства на отделните фази с цел избягване на структурни зависимости.

Конвейерна реализация

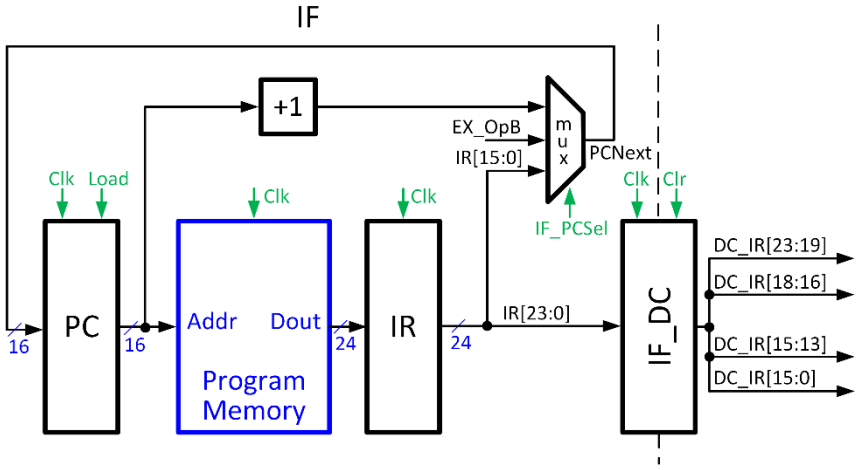
Както е известно, конвейеризацията (pipelining) на инструкциите е апаратен механизъм, който позволява груповото изпълнение на n инструкции с припокриване във времето. Всяка отделна фаза се изпълнява в съответно стъпало на изпълнителния конвейер.

При преобразуването на многоцикловата реализация на процесора в конвейерна отделните функционални устройства се обособяват в самостоятелни конвейерни стъпала, като всяко едно стъпало се използва от точно определена фаза на инструкциите. Допълнително, конвейерните стъпала се ‘развързват’ информационно на входа и изхода си с цел осигуряване на независимост на управляващата и обработваната информация. За целта между стъпалата се разполагат съответни конвейерни регистри.

Дълбочината на конвейера на LCP-R16P е 4, като структурата му включва следните стъпала:

- IF - извличане на инструкцията от програмната памет;
- DC - декодиране на инструкцията, извличане на операндите от регистровия файл;
- EX – изпълнение на операция от ALU, обръщение към паметта за данни;
- WB - обратен запис на резултата в регистровия файл.

Структурата на стъпало IF е показана на Фиг. 2. По адреса, съдържащ се в програмния брояч (PC), от програмната памет се извлича инструкцията и се записва в регистъра на инструкцията (IR).

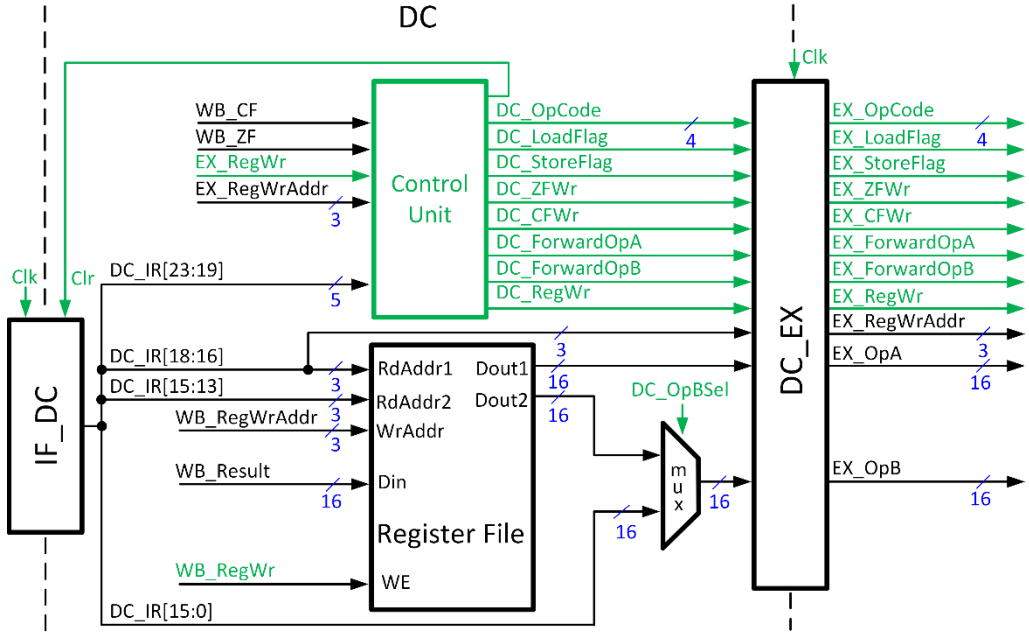


Фиг. 2. Структура на стъпало IF

Информацията, която конвейерният регистър IF_DC пренася към следващото стъпало, включва машинния код на извлечената и записана в IR инструкция. Мултиплексорът, свързан с програмния брояч (PC) е изместен от фаза DC във фаза IF с цел избягване на закъснението, внасяно от инструкция JMP. В този случай при безусловен преход PC се зарежда директно с адреса на прехода, намиращ се в IR[15:0]. Решението за извършване на условен преход (при инструкции JC, JNC, JZ, NZ) се взема във фаза EX поради необходимостта да се изчака установяването на флаговете от предната инструкция. При извършен условен преход PC се зарежда от EX_OpB, който съдържа адресното поле на

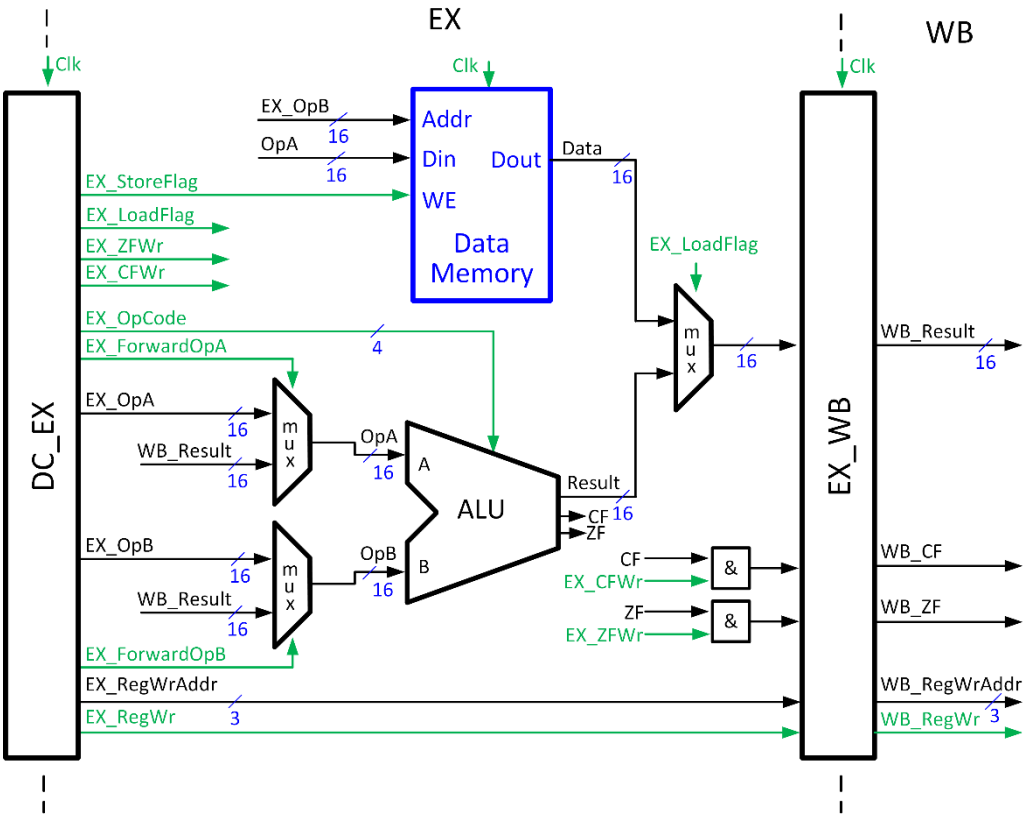
инструкцията във фаза EX. При внасяне на цикъл на изчакване регистър IF_DC се нулира, което съответства на машинния код на инструкция NOP.

Структурата на стъпало DC е визуализирана на Фиг. 3. Управляващото устройство декодира инструкцията чрез анализ на най-старшите 5 бита на IF_DC и генерира контролните сигнали, необходими за обработката на инструкцията в следващите стъпала. В това стъпало се извършва четенето на операндите от регистровия блок.



Фиг. 3. Структура на стъпало DC

Структурата на стъпала EX и WB е показана на Фиг. 4.



Фиг. 4. Структура на стъпала EX и WB

Операндите на ALU (OpA и OpB) се получават съответно от EX_OpA и Ex_OpB, но при необходимост от пренасочване на някой от двата операнда стойността му се взема от WB_Result. В стъпало EX се извършва и обръщението към паметта за данни. Адресът на клетката за достъп се съдържа в полето EX_OpB на конвейерния регистър DC_EX. При инструкция STORE данните за запис се получават от OpA (след евентуално пренасочване), а операцията се управлява от сигнал Ex_StoreFlag. Поле WB_Result на конвейерния регистър EX_WB съдържа резултата за запис в регистровия блок, а WB_RegWrAddr задава номера на регистъра, в който трябва да се запише самият резултат. Флаговете за пренос и нулев резултат WB_CF и WB_ZF се обновяват само ако съответните битове EX_CFWr и EX_ZFWr имат стойност ‘1’. На това стъпало са разположени входният и изходният порт, които не са показани с цел опростяване на схемата.

Зависимости, водещи до нарушаване на конвейера

Както е известно, зависимостите водят до нарушаване нормалната работа на конвейера – следващата инструкция от потока инструкции не може да започне изпълнението на дадена фаза в отделения за това машинен цикъл. Зависимостите се групират в три класа: структурни зависимости, зависимости по данни и зависимости по управление.

Структурните зависимости възникват, когато две или повече инструкции не могат да се изпълняват едновременно в конвейера поради наличие на ресурсен конфликт (недостиг на апаратни ресурси). За избягване на структурните зависимости в конкретния случай се използват отделни памети за програми и данни. Структурната зависимост, произтичаща от едновременното обръщение към регистровия файл във фаза DC (за четене) и във фаза WB (за запис), се предолява чрез използване на два отделни порта за запис и четене в регистрите.

Зависимостите по данни възникват вследствие на промяната на последователността на операциите четене/запис, зададена в програмата. Например, нека разгледаме изпълнението на следния код:

```
ADD R1, R3          ; (I1): R1 <- R1 + R3
SUB R2, R1          ; (I2): R2 <- R2 - R1
ADDI R1, 5          ; (I3): R1 <- R1 + 5
```

Инструкция (I1) записва резултата в R1 по време на фазата си WB, докато (I2) чете R1 по време на фазата си DC, като в края ѝ записва съдържанието му в полето EX_OpA на конвейерния регистър DC_EX. Тъй като този момент предшества фазата WB на (I1), то инструкцията (I2) ще се изпълни със старото съдържание на R1. Следователно фазата DC на (I2) трябва да бъде задържана с един цикъл. Инструкция (I3) винаги ще се изпълни с актуалното съдържание на R1.

За редуциране на броя на непроизводителните цикли вследствие на зависимостите по данни е приложен метода на пренасочването (forwarding). При този метод се добавят алтернативни маршрути за данните и управляваща логика за избор на маршрут. Проверката за наличие на зависимости по данни се извършва във фаза DC. Логиката за детектиране на зависимости формира два флага, предавани към фаза EX: EX_ForwardOpA и EX_ForwardOpB, съответно за пренасочване на операнди OpA и OpB на ALU. Пренасочването работи само когато резултатът от предишната инструкция следва да се запише в регистър. Флагът ForwardOpA е активен, когато номерът на регистъра-приемник на резултата (Rd) от текущата инструкция е идентичен с номера на регистър Rd от предишната. Съответно, флагът ForwardOpB се установява при съвпадение на номера на регистъра източник (Rs) от текущата инструкция с номера на регистър Rd от предишната. За конкретния пример при инструкция (I2) във фаза EX ще се извърши пренасочване на WB_Result=(R1+R3) към втория вход на ALU.

Зависимостите по управление са следствие главно на условните преходи и други инструкции, водещи до промяна на регистър PC. За да се намали загубата на производителност, е реализирана следната стратегия:

- Декодирането на инструкция JMP се извършва още на фаза IF, при което PC се обновява с адреса на прехода преди извличането на следващата инструкция, т.е. няма загуби;
- При декодиране на инструкция за условен преход винаги се извлича и записва в IF_DC следващата я инструкция. В началото на фаза EX на инструкцията за условен преход става ясно дали трябва да се извърши преход или не. Ако преход не трябва да се извърши, се продължава изпълнението на записаната в IF_DC инструкция, т.е. няма загуби. Ако трябва да се извърши преход, съдържанието на IF_DC се нулира, което съответства на запис на машинния код на инструкция NOP. В същия момент програмният брояч се презарежда и в следващия цикъл се стартира изпълнението на инструкцията, която се намира на адреса на прехода, т.е. има загуба от един цикъл.

ЗАКЛЮЧЕНИЕ

Описаният в доклада модел служи за осмисляне както на базовите архитектурни аспекти на процесора, така и на един от основните апаратни механизми за повишаване на производителността – конвейеризацията на инструкциите.

За спецификация, моделиране, симулация и реализация на процесора се използва интегрираната среда за проектиране Xilinx Vivado Design Suite (Vivado, 2023). Тестването се извършва в средата на развойния макет Basys 3 (Digilent, 2016), включващ FPGA схема XC7A35T от фамилията Artix-7. Благодарение на използването на FPGA логика за реализация на процесора студентите имат възможност да правят промени в архитектурата и да провеждат различни експерименти: промяна на разрядността на процесора и броя на регистрите в регистровия блок, добавяне на нови инструкции и методи за адресация и др..

БЛАГОДАРНОСТ

Този доклад се публикува с подкрепата на проект 2025-ФЕЕА-02 „Приложение на AI в различни професионални сфери: възможности, предизвикателства и етични въпроси“, финансиран от фонд „Научни изследвания“ на Русенския университет „Ангел Кънчев“.

REFERENCES

Assumpção, J., Ando, O., de Araújo, H., & Gazziro, M. (2024). *An Educational RISC-V-Based 16-Bit Processor*. Chips, Volume 3 - Issue 4, December 2024, pp. 395-407.

Chen, Lin & Ma, Xiao & Ji, Xiang (2023). FPGA-based LoongArch Five-stage Pipeline CPU. Journal of Physics: Conference Series.

Husainali S., Hitesh N., & Abhishek A. (2016). *Design of 32-bit 3-Stage Pipelined Processor based on MIPS in Verilog HDL and Implementation on FPGA Virtex7*. International Journal of Applied Information Systems. New York, USA, Volume 10 – No. 9, May 2016, pp. 26-37.

Navarro-Torrero, P., Martínez-Rodríguez, M. C., Barriga-Barros, Á., & Brox, P. (2024). *Full Open-Source Implementation of an Academic RISC-V on FPGA*. 2024 XVI Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica (TAEE), Malaga, Spain, 2024, pp. 1-5.

Patterson, D., & Hennessy, J. (2014). *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.