

Programiranje 2, letnji semestar 2023/4

Osnovna struktura C/C++ programa

Stefan Nikolić

Prirodno-matematički fakultet, Novi Sad

26.02.2024.

Na prošlom predavanju smo pokušali da ilustrujemo zašto je za matematiku važno programiranje

Jedan primer u finansijskoj matematici

Bloomberg



Baruch College campus in New York. /Photographer: Victor J. Blue/Bloomberg

School of Quant: At \$29,000, a Public NYC College Outclasses Princeton

Here are the hard numbers: An 18-month Baruch MFE costs a New York City resident about \$29,000. A similar degree from MIT costs four times that - and starting salaries for those grads are about a fifth lower. On average, fresh Baruch MFEs can expect almost \$170,000, before bonus, according to QuantNet. (Baruch puts that figure closer to \$220,000.)



Dan Stefanica

Jedan primer u finansijskoj matematici

The screenshot shows a search result for the book "A Primer for the Mathematics of Financial Engineering" by Dan Stefanica. The book is part of the "Financial Engineering Advanced Background Series". It has a rating of 4.5 stars from 73 ratings. The page also includes reviews and links to other formats.

A Primer For The Mathematics Of Financial Engineering, Second Edition (Financial Engineering Advanced Background Series)

by [Dan Stefanica](#) (Author)

4.5 ★★★★☆ 73 ratings

[Book 1 of 4: Financial Engineering Advanced Background Series](#) [See all formats and editions](#)

Reviews for "A Primer for the Mathematics of Financial Engineering", First Edition:

One of the hottest degrees on today's campus is a Masters in Financial Engineering. Whether you need to retrieve hallowed memories or just want to familiarize yourself with the mathematics underlying this degree, this unique book offers a terrific return on investment."

--Peter Carr, PhD

Global Head of Modeling, Morgan Stanley; Director of the Masters Program in Mathematical Finance, Courant Institute, NYU

This is the book I always recommend to people who ask about their mathematics before

Books Advanced Search New Releases Best Sellers & More Amazon Book Clubs Children's Books Textbooks Best Books of the Month Best Books

Books > Business & Money > Job Hunting & Careers

Financial Engineering Advanced Background Series

A Primer for the Mathematics of Financial Engineering
SECOND EDITION
Dan Stefanica

$\Delta(P_{ATM}) \approx -\frac{1}{2} + 0.2\sigma\sqrt{T}$

$x_{k+1} = x_k - (DF(x_k))^{-1}F(x_k)$

$\Delta V \approx -D_S(V)\delta r + \frac{C_S(V)}{2}(\delta r)^2$

FE Press New York

Jedan primer u finansijskoj matematici

The screenshot shows a web browser window with the URL <https://mfe.baruch.cuny.edu/alumni-testimonials/>. The page content is a testimonial by David Abitbol:

My experience at Baruch was outstanding. The faculty is exceptionally competent and directly connected to the financial industry. I was lucky enough to be part of one of last Salih Neftçi class, who was one of the most gifted teachers and researchers in financial engineering.

The classes were always up to date in terms of current needs in industry: I was able to take the Structured Finance class and make a career out of it very easily due to the high quality of the teaching. Also, the program emphasizes the need for strong programming skills; learning C++ was mandatory which helped tremendously when looking for a job.

Finally the placement support offered at Baruch is extremely professional and that's why I was able to work right after I graduated and became a senior financial engineer at one of the biggest hedge fund in the world.

David Abitbol, MFE'05, Senior Financial Engineer, Hedge Fund Industry



Jedan primer u finansijskoj matematici

← → ⌂

https://www.linkedin.com/in/abitboldavid/

 Search

Home My Network Jobs Messaging Notifi



David Abitbol · 3rd
Senior Financial Engineer at Davidson Kempner Capital Management
Austin, Texas, United States · [Contact info](#)
500+ connections

[Message](#) [+ Follow](#) [More](#)

 **Davidson Kempner Capital Management**
 **City University of New York-Baruch College**

Jedan primer u finansijskoj matematici

https://en.wikipedia.org/wiki/Davidson_Kempner_Capital_Management 120%

Davidson Kempner Capital Management

1 language ▾

Article [Talk](#)

[Read](#) [Edit](#) [View history](#) [Tools](#) ▾

From Wikipedia, the free encyclopedia

Davidson Kempner Capital Management LP (often known by the short form **Davidson Kempner**) is a global institutional [alternative investment](#) management firm with over \$36 billion in assets under management.^{[5][6]}

Davidson Kempner is headquartered in New York City, with additional offices in London, Hong Kong, Dublin, Philadelphia, Shenzhen and Mumbai.^{[7][8][9]}

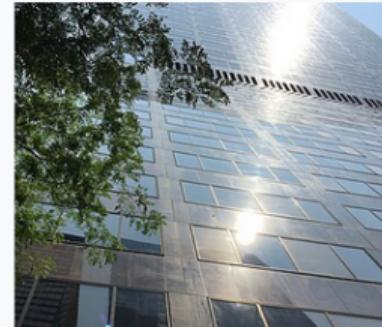
The firm is led by Anthony A. Yoseloff who serves as Executive Managing Member and Chief Investment Officer.^[10]

As of June 30, 2022, Davidson Kempner was ranked as the 8th largest [hedge fund](#) in the world.^[11]

Davidson Kempner has approximately 500 employees in the firm's seven offices.^[4] The firm is headquartered at 520 Madison Avenue in New York, New York.^{[5][12]}

Davidson Kempner Capital Management LP

DavidsonKempner
Capital Management LP



Jedan primer u finansijskoj matematici

The screenshot shows a web browser window with the URL <https://www.maxeler.com/solutions/financial-analytics/>. The page header features the Maxeler Technologies logo with the tagline "Maximum Performance Computing". The navigation menu includes links for Platforms, Products, Technology, and About. Below the menu, a secondary navigation bar lists Oil & Gas, Financial Analytics (which is highlighted in blue), Low Latency, Scientific Computing, and University Pro. The main content area has a title "Financial Analytics Platforms" and a quote from Peter Cherasia, Head of Markets Strategies at J.P. Morgan, enclosed in blue double quotes.

Financial Analytics Platforms

With the new Maxeler technology, J.P. Morgan's trading businesses can now compute orders of magnitude more quickly, making it possible to improve our understanding and control of the profile of our complex trading risk.

Peter Cherasia, Head of Markets Strategies, J.P. Morgan

Jedan primer u finansijskoj matematici

The screenshot shows a web browser window with the URL <https://www.dmi.uns.ac.rs/wp-content/uploads/2022/12/Maxeler-is-hiring-C.pdf>. The page title is "MAXELER | MAXELER IS HIRING!". The main text reads: "We are seeking people who care about how computers work with skills in science, programming and High-Performance Computing, to join our team in Belgrade, Novi Sad or Kragujevac. To apply, email your CV to hiring.belgrade@maxeler.com". Below this is an image of a black Groq technology chip module. To the right is the text "JOIN THE DATAFLOW COMPUTING FUTURE!".

MAXELER | MAXELER IS HIRING!

We are seeking people who care about how computers work with skills in science, programming and High-Performance Computing, to join our team in **Belgrade, Novi Sad or Kragujevac**.

To apply, email your CV to hiring.belgrade@maxeler.com.



JOIN THE DATAFLOW COMPUTING FUTURE!

Zadaci za danas

1. Osnovna struktura C/C++ programa
2. Kako radi računar?
3. Šta radi kompjajler?

Zašto je bitno da znamo kako radi računar?

- Da bismo što manje toga morali da učimo napamet
(ako budemo razumeli kako radi računar, nećemo znati samo kako izgledaju programi u C/C++-u, već i zašto tako izgledaju)
- Jer ćemo samo ako razumemo kako radi računar moći da pišemo programe visokih performansi, što je za matematičke probleme jako bitno

Nekoliko napomena

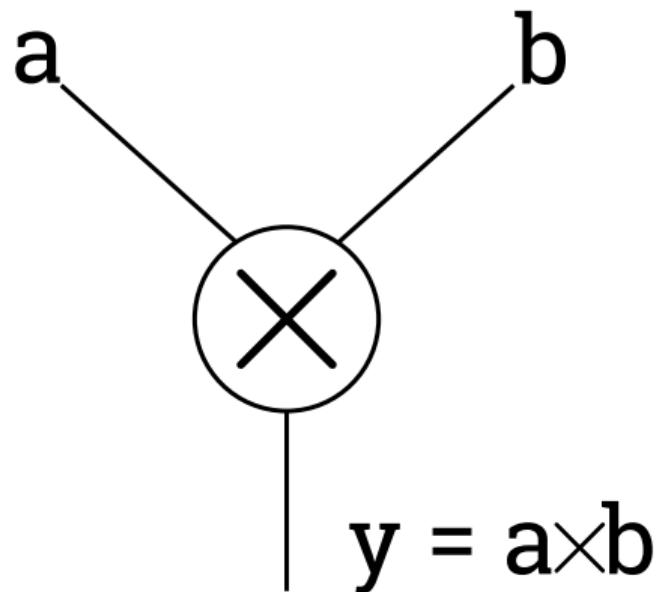
- Obradićemo samo osnovne principe, ali ćemo ih izvesti od početne intuicije
- Na kraju ćemo navesti preporučenu literaturu za one koji budu želeli da znaju više
- Ovo neće ići na usmeni ispit, ali će omogućiti lakše i dublje razumevanje daljeg gradiva

Problem sa prethodnog predavanja: Ojlerova hipoteza o zbirovima stepena

1. Izlistavamo redom (leksikografski) kombinacije $n - 1$ brojeva iz $[1, k]$, $k \in \mathbb{N}$, a_1, a_2, \dots, a_{n-1}
2. Za svaku kombinaciju izračunamo $S = \sum_{i=1}^{n-1} a_i^n$
3. Prepostavimo da je S oblika b^n
4. Izračunamo b kao $\lfloor \sqrt[n]{S} \rfloor$
5. Ako je $S = b^n$, našli smo kontraprimer

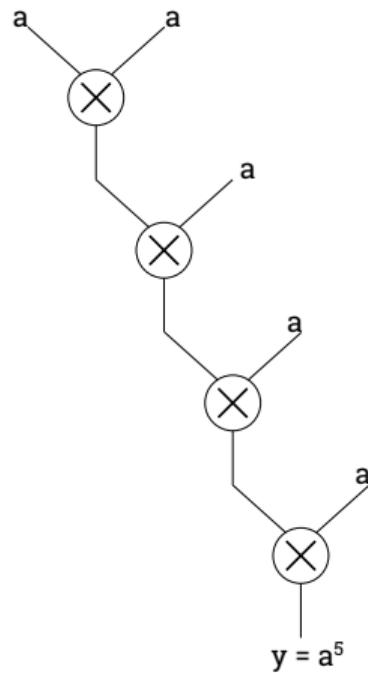
Koja operacija je odnosila najviše vremena?

Množač

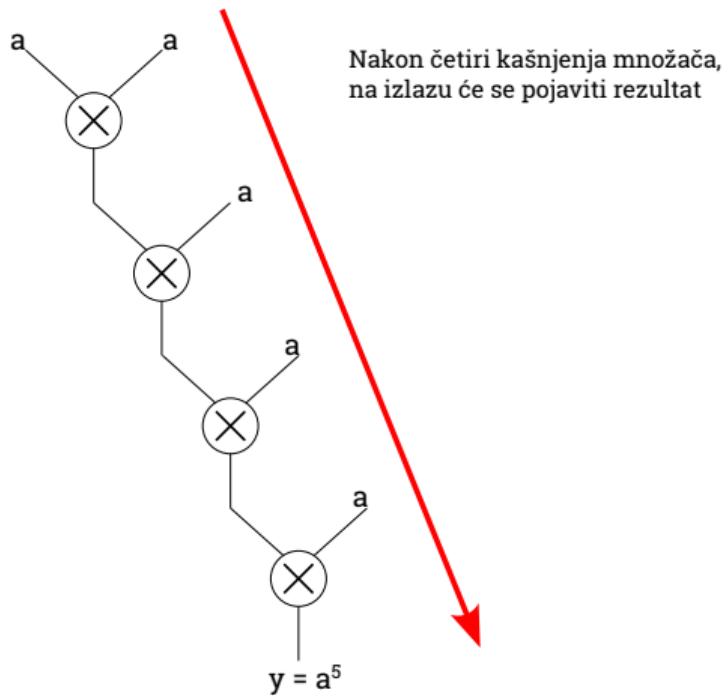


Možemo li da izračunamo a^5 pomoću 4 množača?

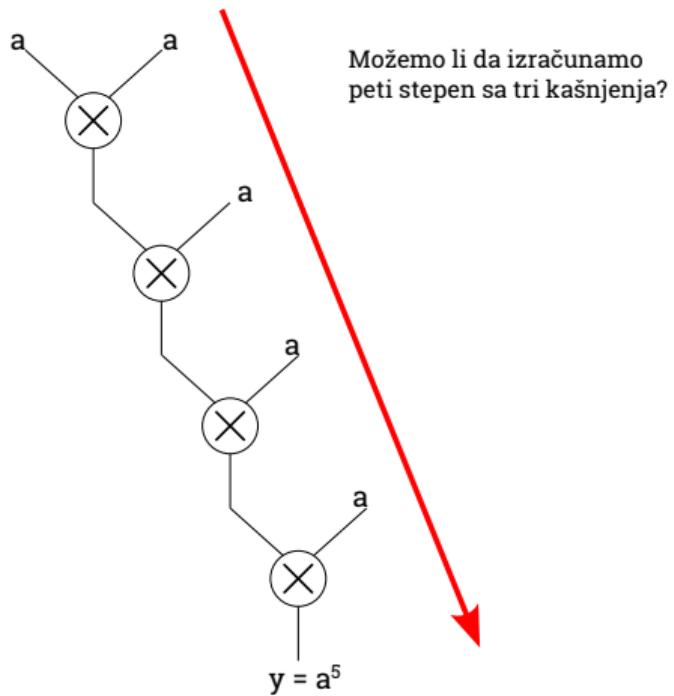
Stepenovanje množacima



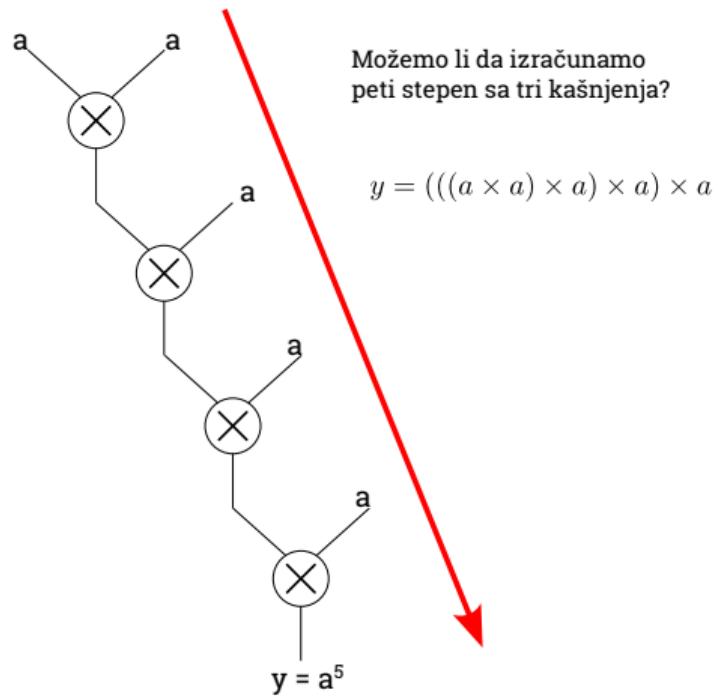
Stepenovanje množićima



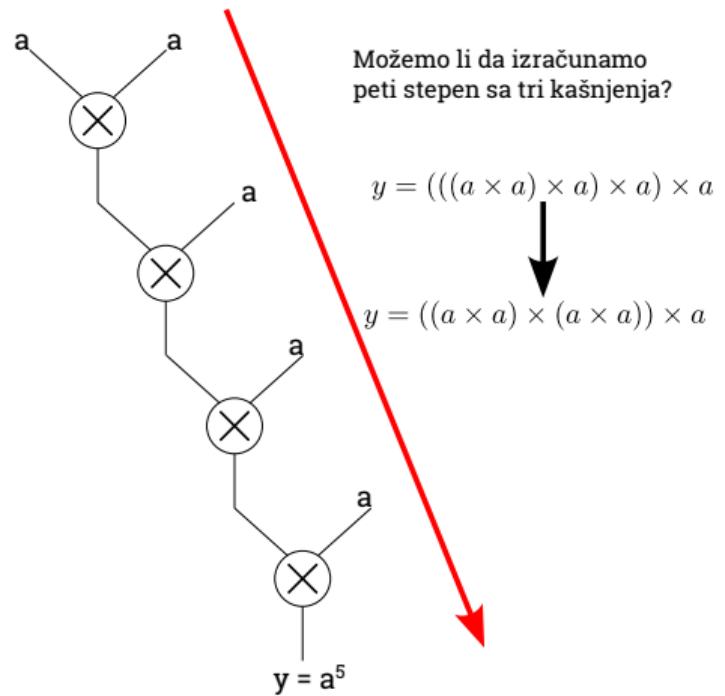
Stepenovanje množićima



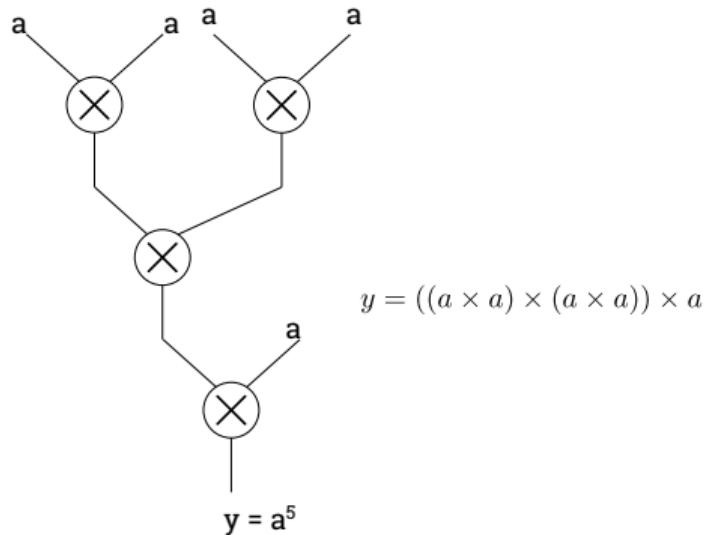
Stepenovanje množićima



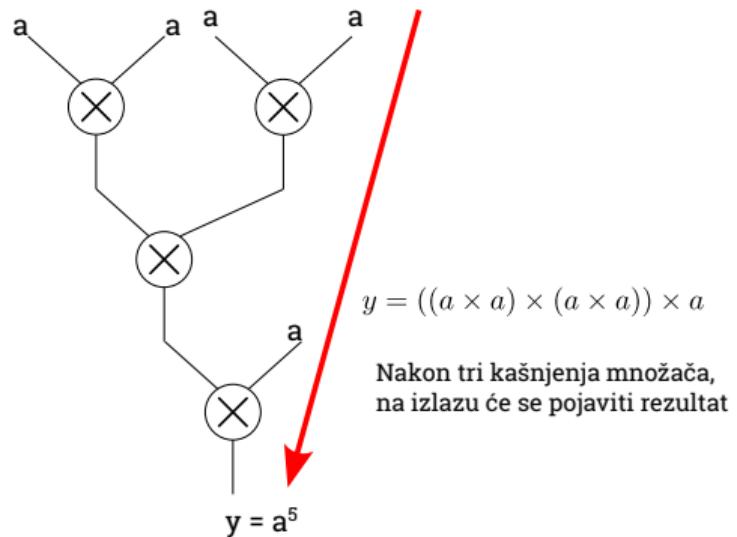
Stepenovanje množićima



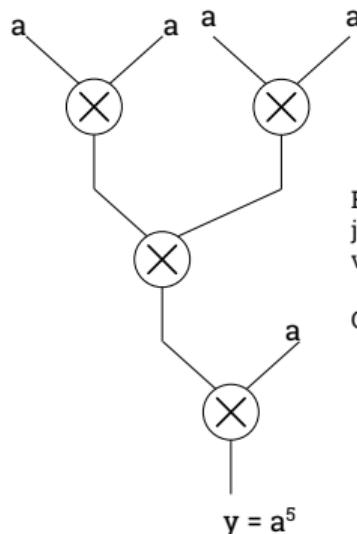
Stepenovanje množacima



Stepenovanje množićima



Stepenovanje množićima



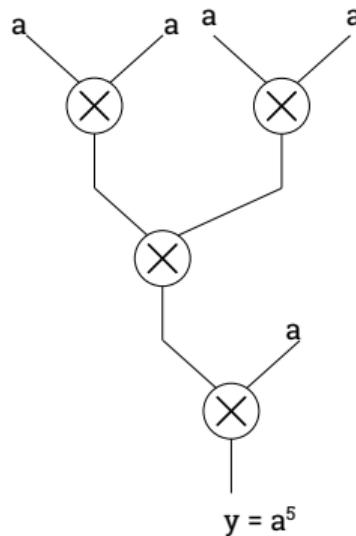
Redukcija visine stabla računanja
je jedna od optimizacija koda koje
vrše optimizujući kompjajleri

O tome više uskoro

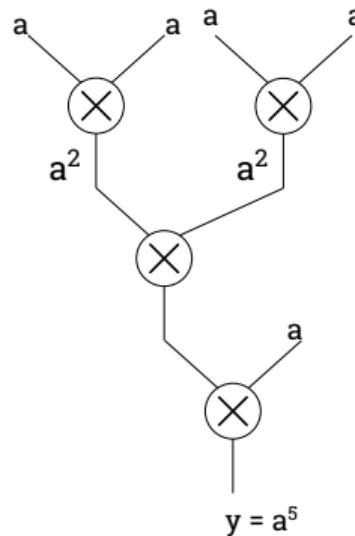
Stepenovanje množaćima

Da li su nam dovoljna tri množaća?

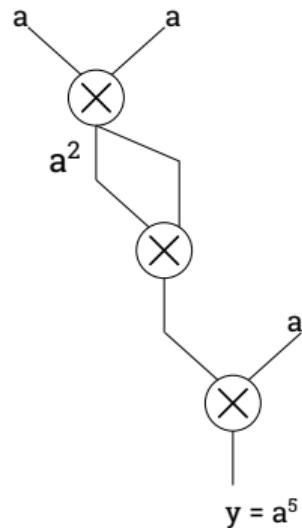
Stepenovanje množacima



Stepenovanje množacima



Stepenovanje množacima



ENIAC



Ruth Licherman and Marlyn Wescoff reprogramming ENIAC in 1946

ENIAC je bio programiran povezivanjem funkcionalnih jedinica pomoću žica kako bi se oformilo kolo koje evaluira željenu funkciju.
Inače, svi prvi programeri su bili žene.

High Frequency Trading Acceleration using FPGAs

Christian Leber, Benjamin Geib, Heiner Litz

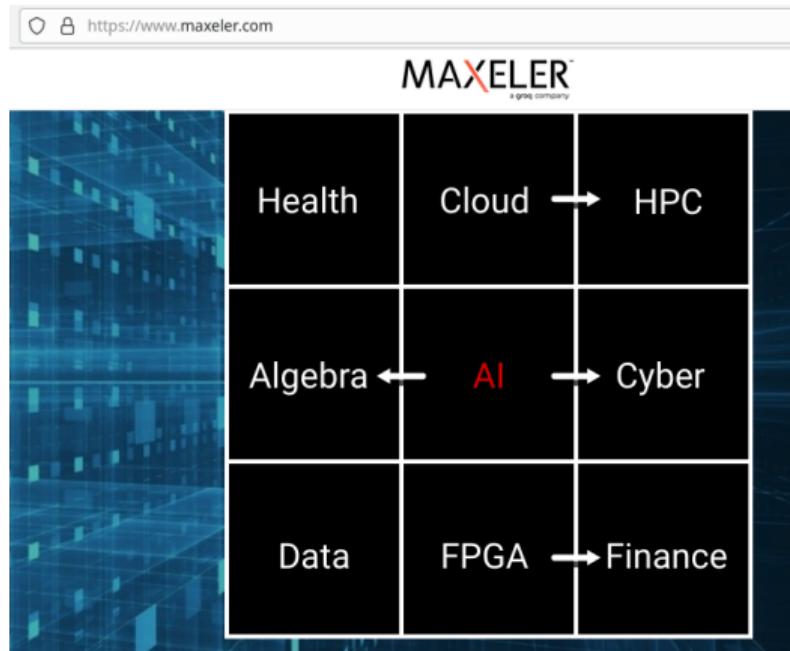
University of Heidelberg

68131 Mannheim, Germany

{christian.leber, benjamin.geib, heiner.litz}@ziti.uni-heidelberg.de

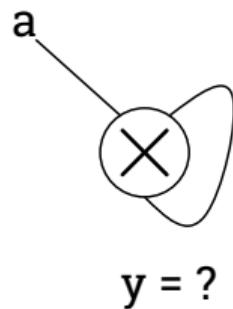
Povezivanje funkcionalnih jedinica vezama na takozvanim FPGA čipovima i do danas ostaje osnova prostornog računarstva (eng. *Spatial Computing*) koje, zbog vrlo malog kašnjenja, ima značajnu primenu u finansijama

Prostorni računari



Možemo li da izračunamo peti stepen koristeći samo jedan množač?

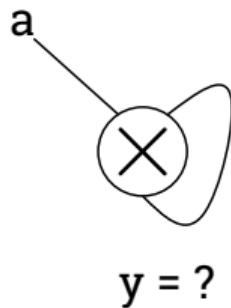
Vremenski multipleks



$$y = ?$$

Da li će se na izlazu u nekom trenutku pojaviti a^5 ?

Vremenski multipleks



Šta nam nedostaje da bismo to mogli da garantujemo?

Vremenski multipleks

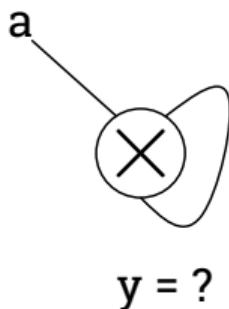


Moramo da obezbedimo poznato početno stanje ulaza množača
(na primer, $a^0 = 1$)

Bitna napomena

U računarstvu najčešće nije moguće prepostaviti da će neki signal/promenljiva imati neku početnu vrednost, već ju je potrebno eksplicitno inicijalizovati!

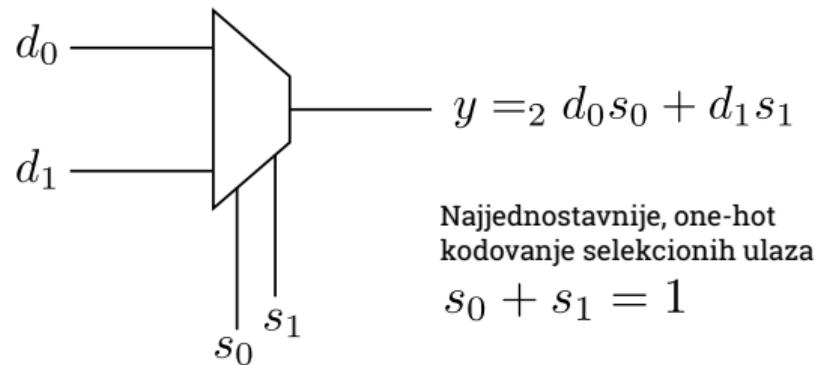
Vremenski multipleks



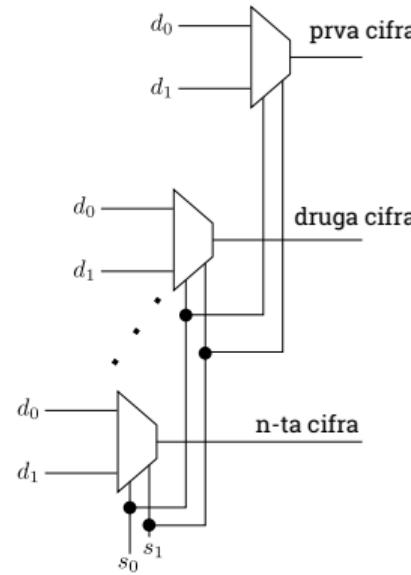
Dakle, potrebno je da na početku na desni ulaz množača dovedemo $a^0 = 1$, a da mu zatim prosledimo izlaz množača

Vreme je da uvedemo još jednu komponentu: multiplekser

Koristićemo $=_2$ da označimo izraze u \mathbb{Z}_2



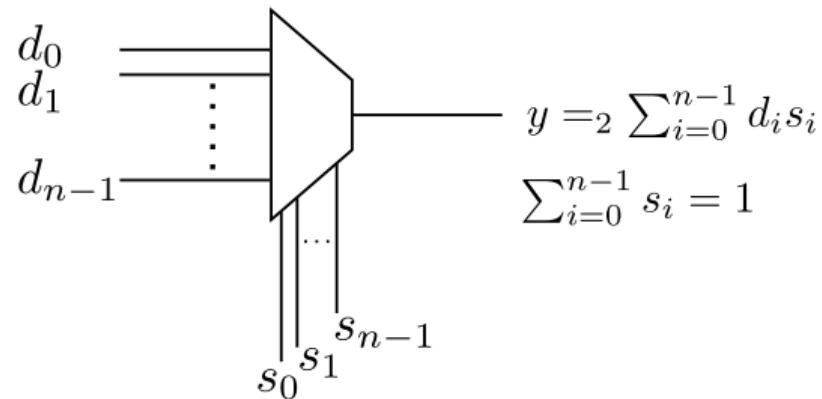
Multipleksiranje višecifrenih binarnih brojeva



Naravno, možemo multipleksirati i višecifrene binarne brojeve: za svaku binarnu cifru dodamo još jedan multiplekser koji deli selekciione ulaze sa svim ostalim pojedinačnim multiplekserima

Multipleksiranje više ulaznih signala

Kao i, na primer, vektorske prostore, i multipleksere možemo generalizovati na više ulaza, pri čemu se u svakom trenutku tačno jedan ulaz preslikava na izlaz



Claude Shannon

文 A 66 languages ▾

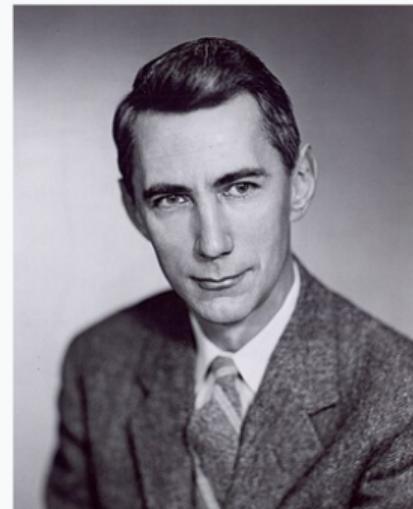
Article Talk

Read Edit View history Tools ▾

From Wikipedia, the free encyclopedia

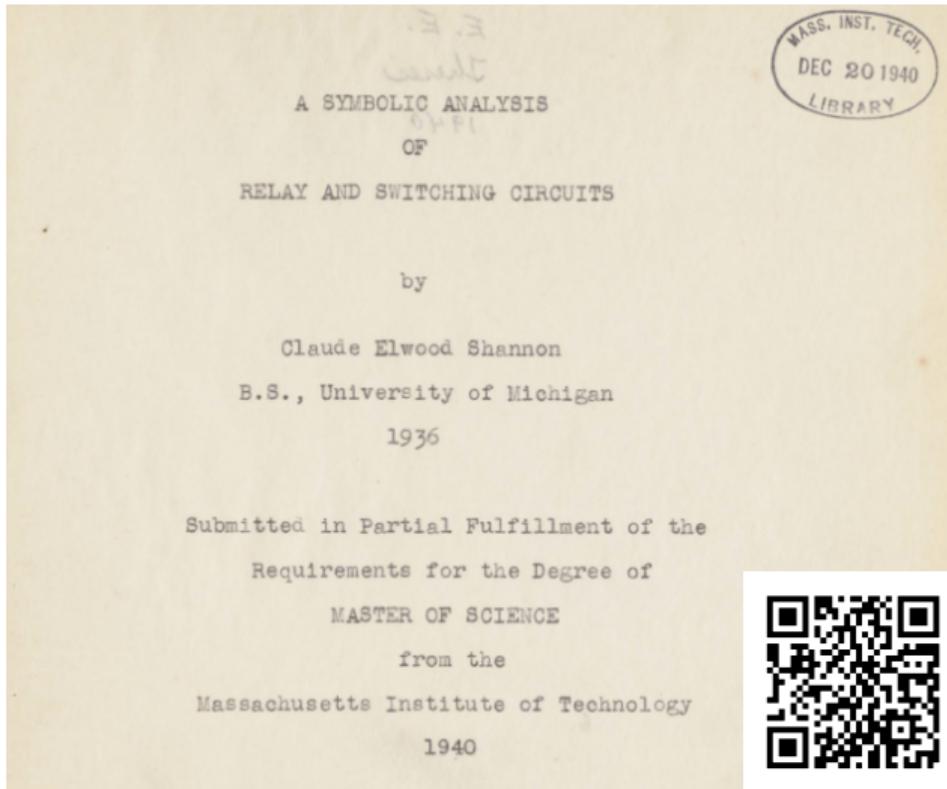
Claude Elwood Shannon (April 30, 1916 – February 24, 2001) was an American [mathematician](#), [electrical engineer](#), [computer scientist](#) and [cryptographer](#) known as the "father of [information theory](#)". He was the first to describe the boolean gates (electronic circuits) that are essential to all digital electronic circuits, and he built the first machine learning device, thus founding the field of [artificial intelligence](#).^{[1][2][3][4]} He is credited alongside George Boole for laying the foundations of the [Information Age](#).^{[5][6][7][4]}

As a 21-year-old [master's degree](#) student at the [Massachusetts Institute of Technology](#) (MIT), he wrote [his thesis](#) demonstrating that electrical applications of Boolean algebra could construct any logical numerical relationship,^[8] thereby establishing the theory behind [digital computing](#) and [digital circuits](#).^{[9][10]} In 1987, [Howard Gardner](#) called his thesis "possibly the most important, and also the most famous, master's thesis of the century",^[11] and [Herman Goldstine](#) described it as "surely ... one of the most important master's theses ever written ... It helped to change digital circuit



Shannon c. 1950s

Srećom, zahvaljujući matematičarima znamo kako da u hardveru implementiramo bilo koju Bulovu funkciju



Srećom, zahvaljujući matematičarima znamo kako da u hardveru implementiramo bilo koju Bulovu funkciju



When I came to Bell Labs, I came into a very productive department. Bode was the department head at the time; Shannon was there, and there were other people. I continued examining the questions, "Why?" and



Richard Hamming

``You and Your Research''

Transcription of the
Bell Communications Research Colloquium Seminar
7 March 1986

Srećom, zahvaljujući matematičarima znamo kako da u hardveru implementiramo bilo koju Bulovu funkciju

АЛГЕБРА ДВУХПОЛЮСНЫХ СХЕМ, ПОСТРОЕННЫХ
ИСКЛЮЧИТЕЛЬНО ИЗ ДВУХПОЛЮСНИКОВ (АЛГЕБРА А-СХЕМ)[†]

Канд. физ.-мат. наук В. И. Шестаков

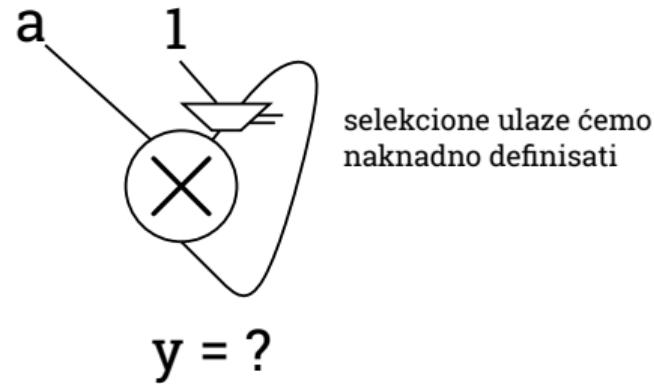
ВЗАИМНО-ОДНОЗНАЧНОЕ СООТВЕТСТВИЕ МЕЖДУ А-СХЕМАМИ И
А-ВЫРАЖЕНИЯМИ
А-схемы



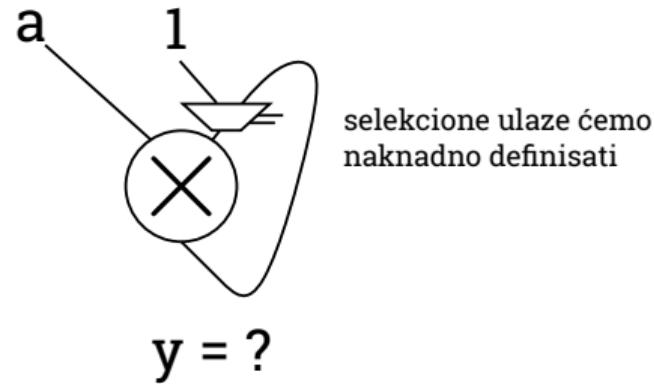
Viktor Ivanovič Šestakov, 1935.
(u pripremi doktorata na Lomonosovu)

Дата рождения	15 октября 1907
Место рождения	Москва, Российская империя
Дата смерти	3 мая 1987 (79 лет)
Место смерти	Москва, СССР
Страна	Российская империя СССР
Научная сфера	математика
Место работы	ИТМИВТ, МГУ
Альма-матер	МГУ
Учёное звание	профессор
Научный руководитель	Гливенко, Валерий Иванович

Da li će se sada na izlazu množača u nekom trenutku pojaviti a^5 ?

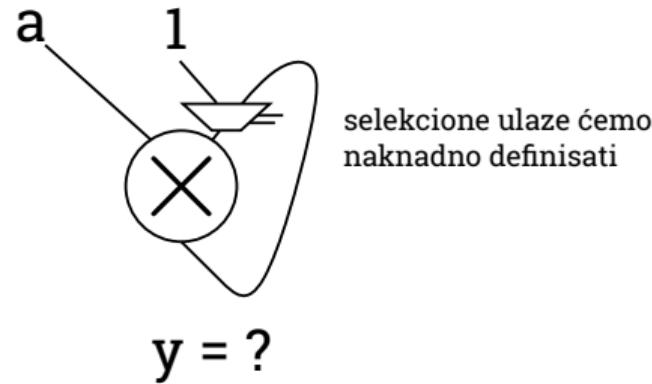


Da li će se sada na izlazu množača u nekom trenutku pojaviti a^5 ?



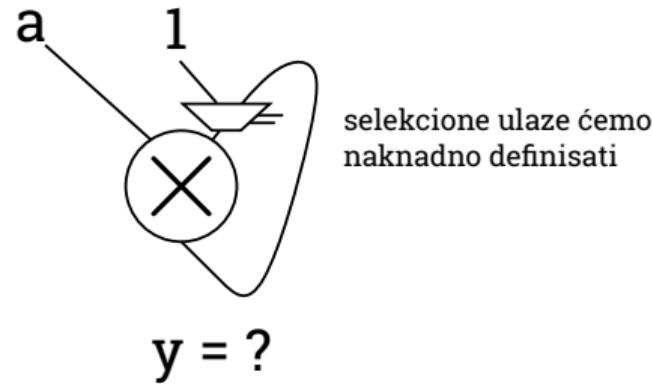
Hoće, ali će ga ubrzo zameniti a^6 , pa a^7 ...

Da li će se sada na izlazu množača u nekom trenutku pojaviti a^5 ?



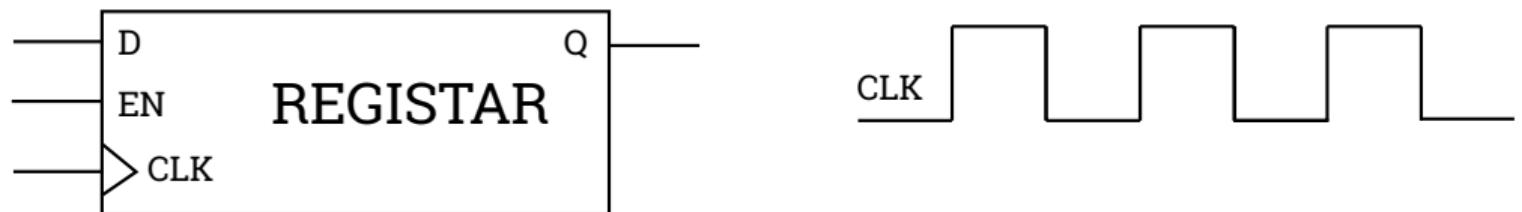
Šta nam onda još nedostaje?

Da li će se sada na izlazu množača u nekom trenutku pojaviti a^5 ?



Moramo nekako da izbrojimo 5 množenja i onda zaustavimo dalje množenje

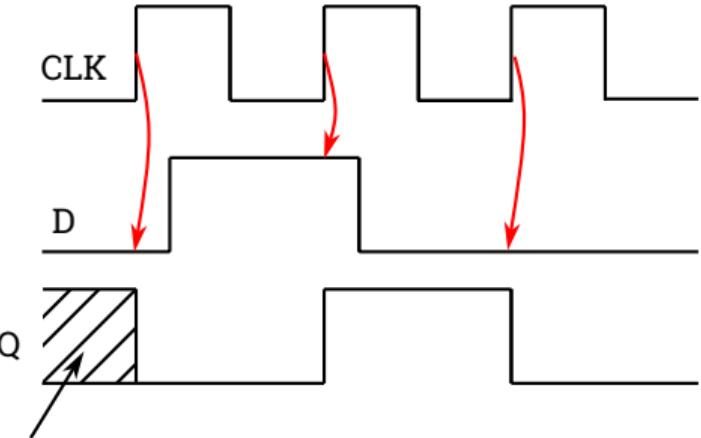
Vreme je da uvedemo još jednu komponentu: registar



Vreme je da uvedemo još jednu komponentu: registar



Kada nađe rastuću ivicu takta,
registar proverava vrednost ulaza D

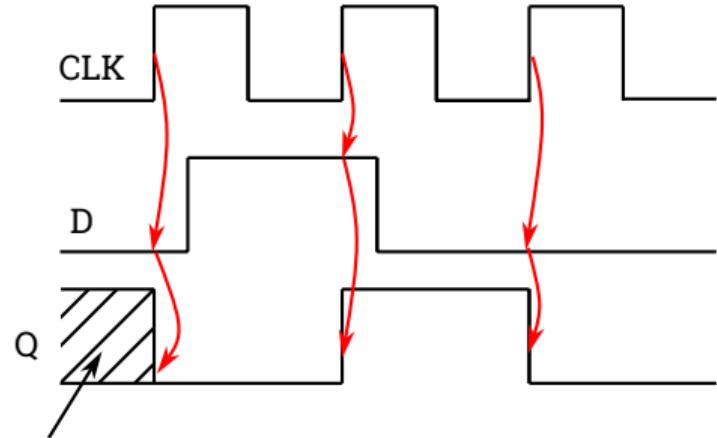


na početku register sadrži nepoznatu vrednost
(zato inicijalizujemo promenljive)

Vreme je da uvedemo još jednu komponentu: registar



Kada nađe rastuću ivicu takta,
registar proverava vrednost ulaza D
i preslikava je na izlaz Q



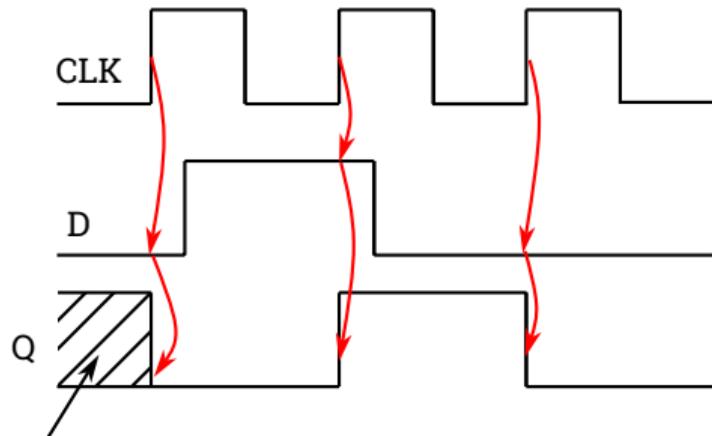
na početku registar sadrži nepoznatu vrednost
(zato inicijalizujemo promenljive)

Vreme je da uvedemo još jednu komponentu: registar



Kada najđe rastuća ivica takta,
registar proverava vrednost ulaza D
i preslikava je na izlaz Q

U međuvremenu, Q zadržava prethodnu vrednost



na početku register sadrži nepoznatu vrednost
(zato inicijalizujemo promenljive)

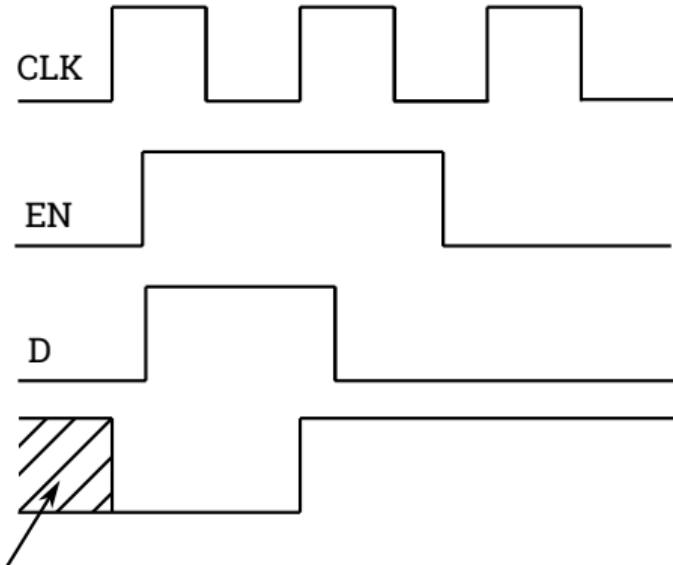
Vreme je da uvedemo još jednu komponentu: registar



EN je signal za dozvolu upisa

Kada je 1, registar normalno funkcioniše

Kada je 0, Q ne menja vrednost, bez obzira na takt

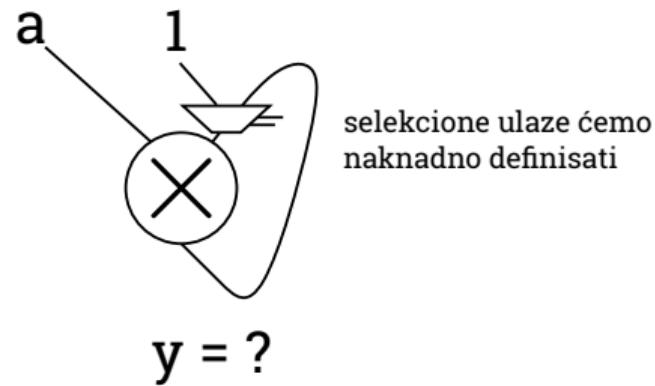


na početku registar sadrži nepoznatu vrednost
(zato inicijalizujemo promenljive)

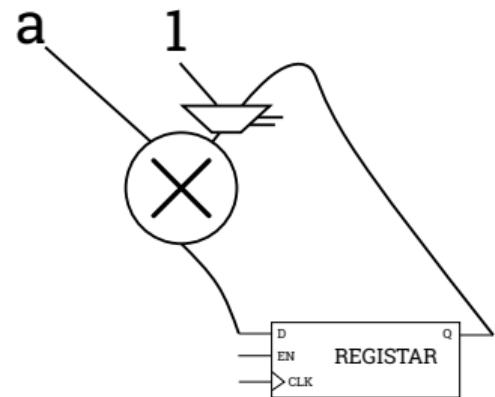
Ukratko o registrima

- Registar je osnovni višebitni memorijski element
- Signal takta diskretizuje vreme
- Registar sprečava „protrčavanje” vrednosti
- Signal EN omogućuje čuvanje vrednosti i kada najđe rastuća ivica takta

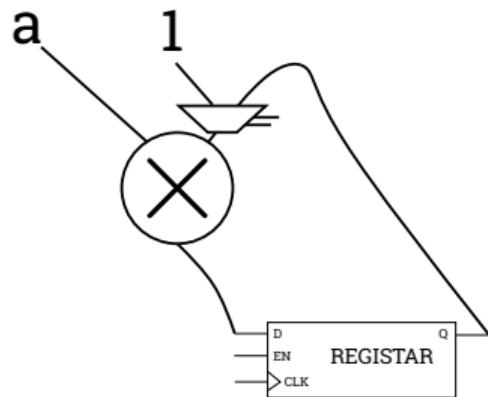
Gde je potrebno da ubacimo registar da bismo sprečili protrčavanje viših stepena?



Gde je potrebno da ubacimo registar da bismo sprečili protrčavanje viših stepena?

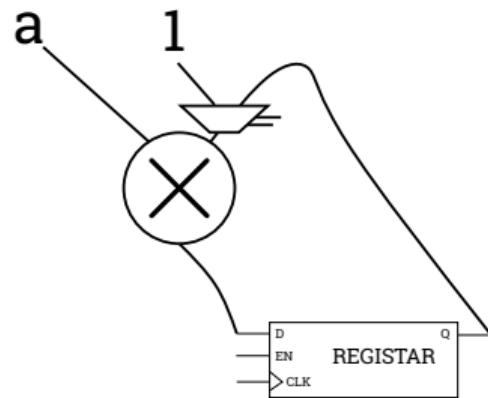


Gde je potrebno da ubacimo registar da bismo sprečili protrčavanje viših stepena?



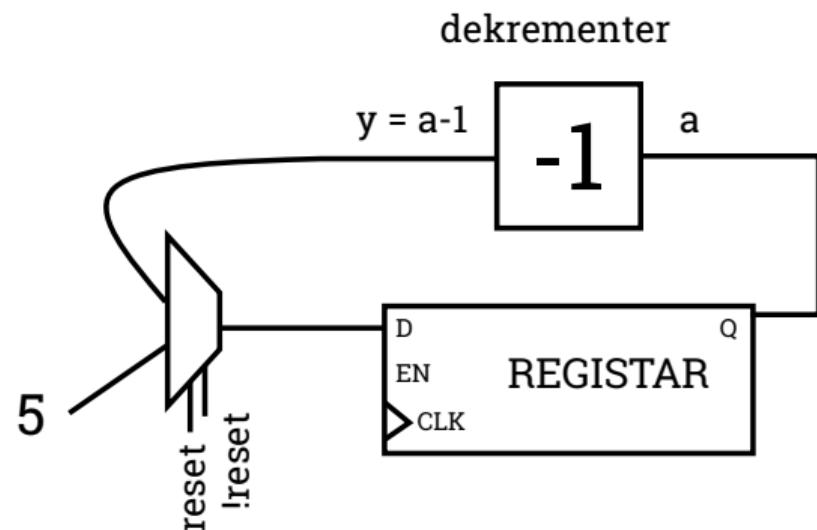
Šta nam još fali?

Gde je potrebno da ubacimo registar da bismo sprečili protrčavanje viših stepena?

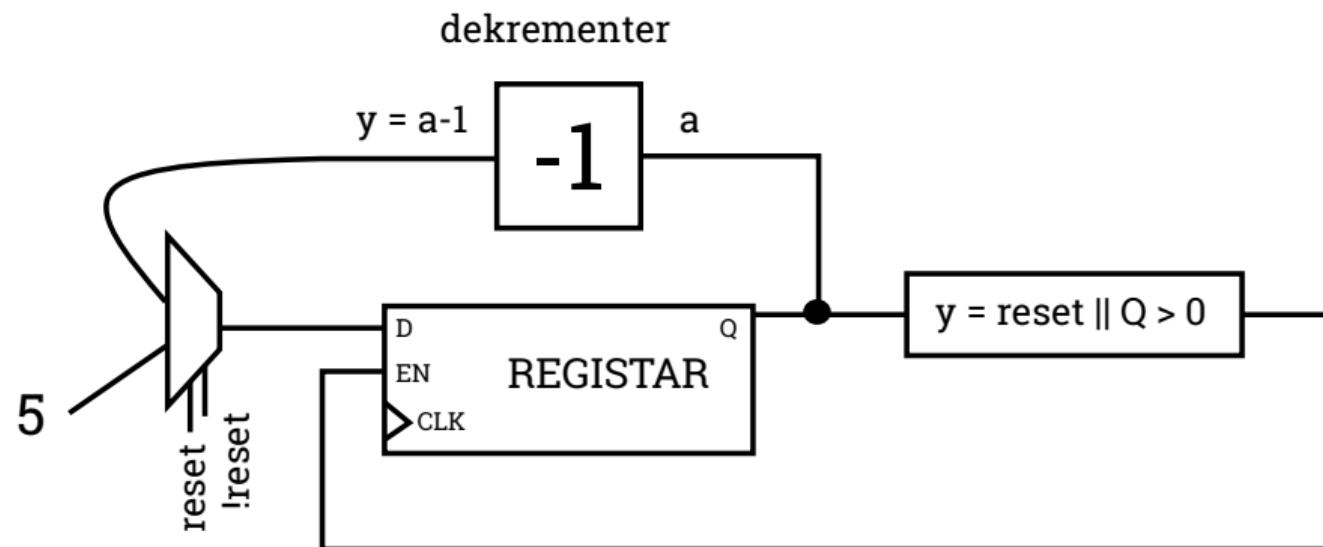


Brojač množenja, da bismo ispravno selektovali ulaze množača i da bismo zaustavili množenje na vreme

Brojač već znamo da napravimo



Brojač već znamo da napravimo



Brojač

Brojač je specijalan slučaj konačnog automata (eng. Finite State Machine). Konačne automate su takođe izmislili matematičari i danas predstavljaju fundamentalni model za sintezu hardvera, ali se često koriste i u programiranju.

The screenshot shows the Wikipedia article for George H. Mealy. The page title is "George H. Mealy". The main content starts with a brief biography: "George H. Mealy (December 31, 1927 – June 21, 2010 in Scituate, Massachusetts)^[1] was an American mathematician and computer scientist who invented the namesake Mealy machine, a type of finite state transducer. He was also a pioneer of modular programming,^{[2][3]} one of the lead designers of the IPL-V programming language,^[4] and an early advocate of macro processors in assembly language programming.^[5]" Below this, there is a section about his work at Harvard University and Bell Laboratories.

George H. Mealy

Contents hide

(Top)

Selected publications

References

3 languages ▾

Read Edit View history Tools ▾

From Wikipedia, the free encyclopedia

George H. Mealy (December 31, 1927 – June 21, 2010 in Scituate, Massachusetts)^[1] was an American mathematician and computer scientist who invented the namesake [Mealy machine](#), a type of finite state transducer. He was also a pioneer of [modular programming](#),^{[2][3]} one of the lead designers of the [IPL-V](#) programming language,^[4] and an early advocate of [macro processors](#) in assembly language programming.^[5]

Mealy went to Harvard University, where he was active in [radio](#) as business manager for [WHRB](#).^[6] He graduated in 1951 with an A.B., and at that time began working for Bell Laboratories.^[7] He later worked at the [RAND Corporation](#)^[8] then [IBM](#)^[9] and taught at Harvard.^[10]

The screenshot shows the Wikipedia article for Edward F. Moore. The page title is "Edward F. Moore". The main content starts with a brief biography: "For other people named Edward Moore, see [Edward Moore \(disambiguation\)](#)". Below this, there is a section about his work as a professor of mathematics and computer science.

Edward F. Moore

Contents hide

(Top)

Biography

Scientific work

Publications

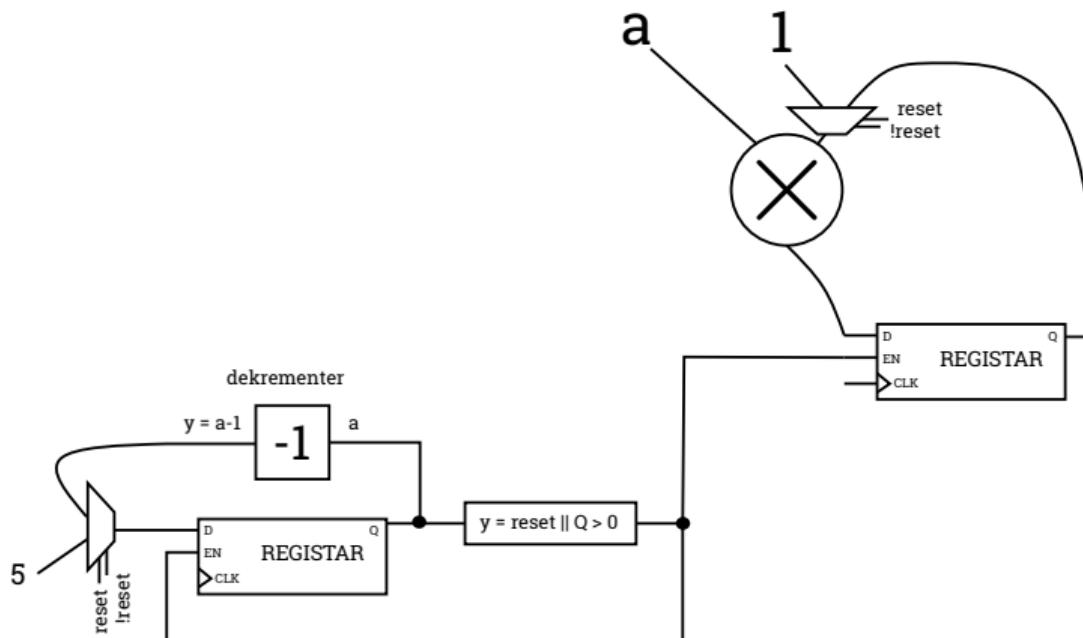
See also

From Wikipedia, the free encyclopedia

For other people named Edward Moore, see [Edward Moore \(disambiguation\)](#).

Edward Forrest Moore (November 23, 1925 in Baltimore, Maryland – June 14, 2003 in Madison, Wisconsin) was an American professor of [mathematics](#) and [computer science](#), the inventor of the Moore finite state machine, and an early pioneer of artificial life.

Kako izgleda celo kolo za stepenovanje jednim množačem?



Šta smo dobili ovim?

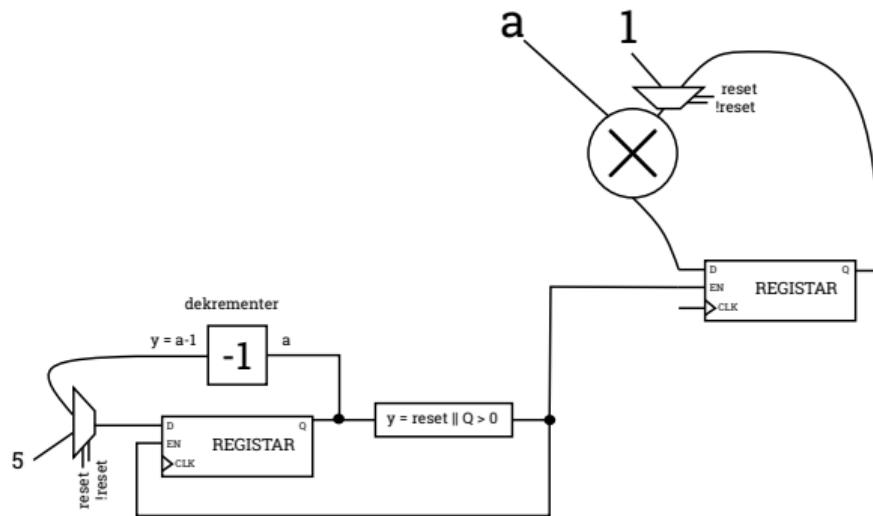
- Umesto tri, koristimo samo jedan množač

U vreme nastanka ENIAC-a, aritmetički sklopovi poput množača su bili izuzetno skupi za realizaciju, pa je smanjenje njihovog broja bilo od velikog značaja

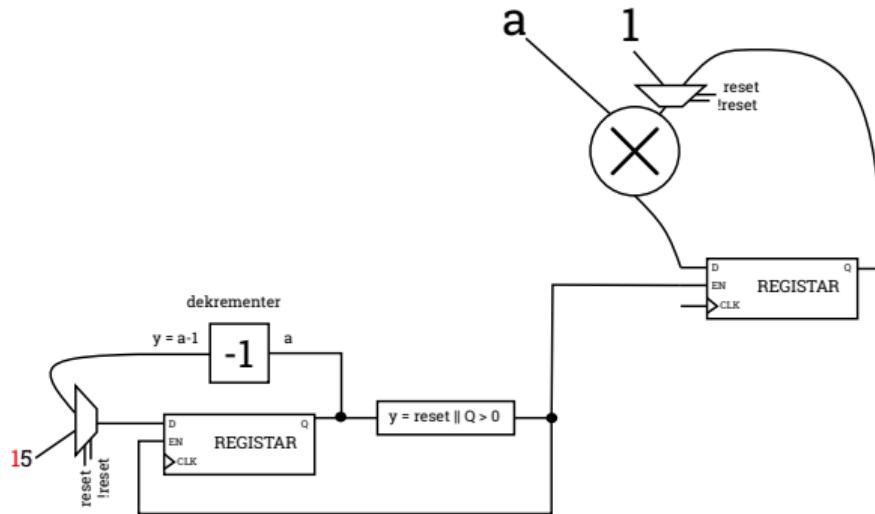
Šta smo još dobili?

Kako bi izgledalo kolo za računanje 15. stepena?

Šta smo još dobili?



Šta smo još dobili?



Više ne moramo da ponovo povezujemo komponente kako bismo izračunali neki drugi stepen. Potrebno je samo da neku drugu vrednost upišemo u odgovarajući registar. To je ključni koncept „vremenskog računanja“ (eng. *Temporal Computing*) koje je danas dominantno. U vreme ENIAC-a, povezivanje je iziskivalo ručni rad, pa je promena paradigme i iz tog razloga bila veoma važna. Na današnjim FPGA čipovima, dešava se automatski.

Uopštavanje

Kako da računamo nešto složenije od stepena?

Uzmimo za primer izraz oblika $\sum_{i=0}^{N-1} a_i^{n_i}$ i jednu konkretnu instancu $a_0^3 + a_1^2 + a_2^4 + a_3$

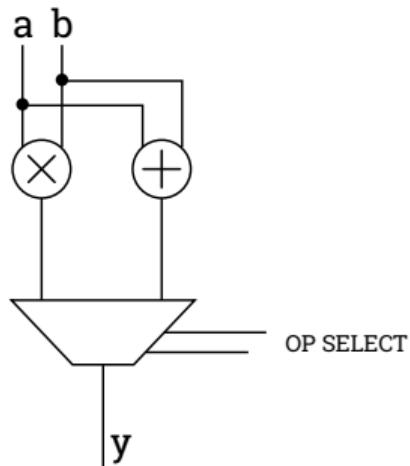
Koje su nam operacije potrebne?

- sabiranje
- množenje

Šta smo do sad naučili?

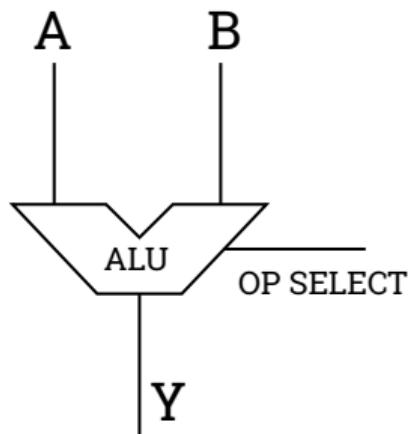
- Možemo da implementiramo bilo koju Bulovu funkciju
- To uključuje i multiplekser kao specijalan slučaj

ALU

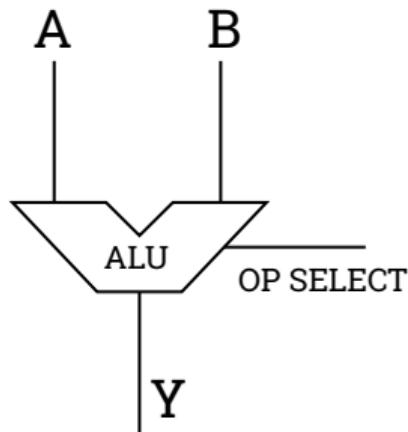


ALU

Naravno, možemo uključiti i više od dve operacije. U opštem slučaju, ovakav blok nazivamo „aritmetičko-logičkom jedinicom“ (eng. *Arithmetic Logic Unit (ALU)*)

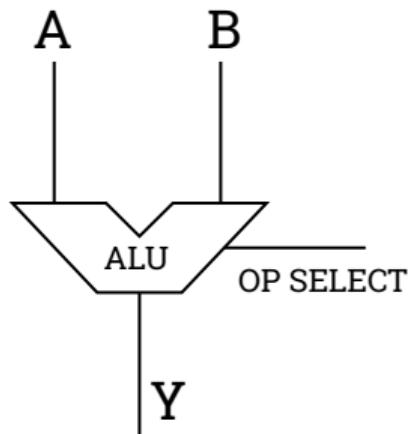


ALU



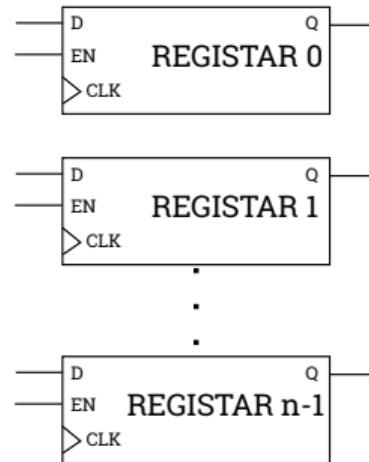
Da bismo izračunali $a_0^3 + a_1^2 + a_2^4 + a_3$, šta je potrebno da menjamo u vremenu?

ALU

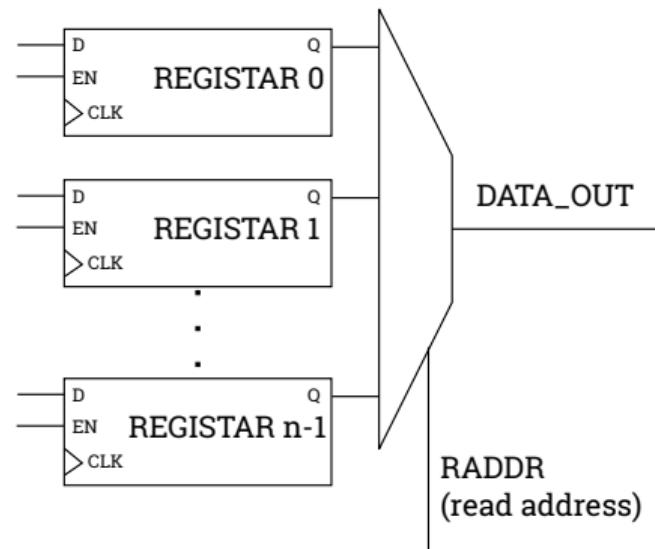


Potrebno je da menjamo operaciju i operande

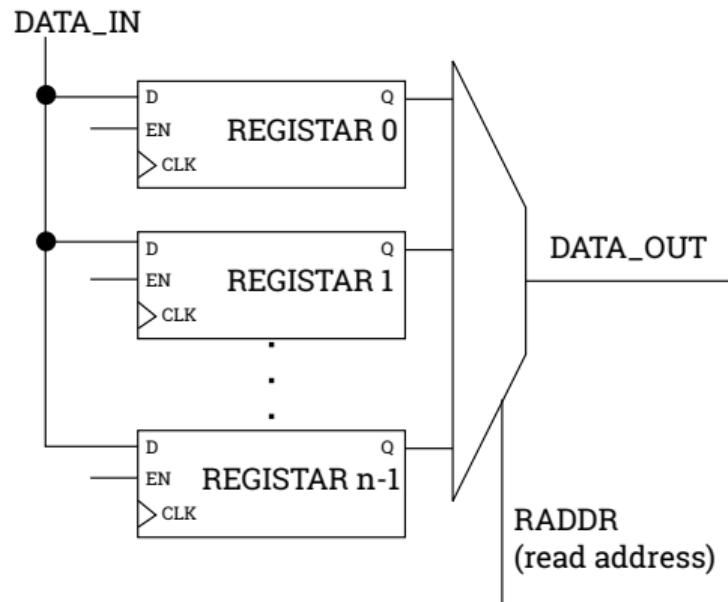
Ponovo slažemo kockice: registrska banka



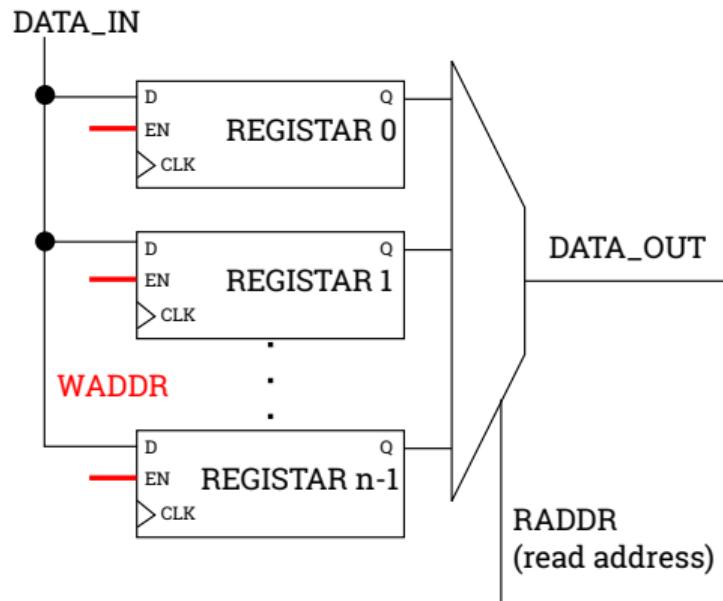
Ponovo slažemo kockice: registrska banka



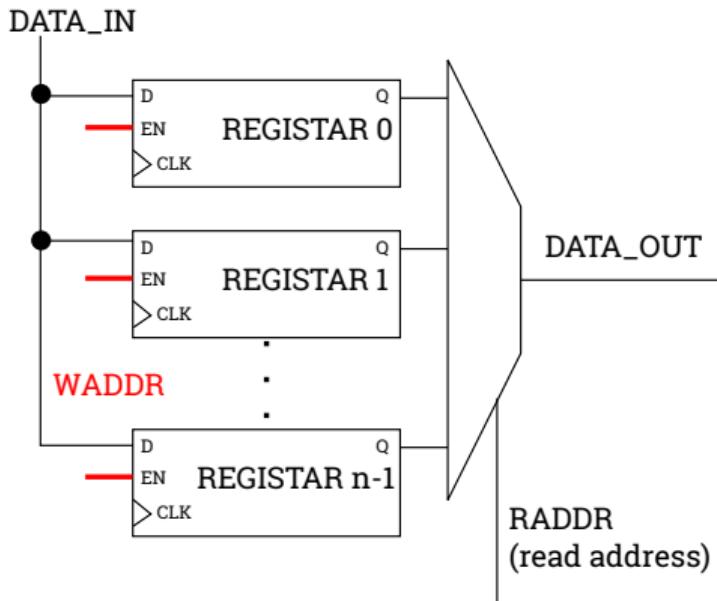
Ponovo slažemo kockice: registrska banka



Ponovo slažemo kockice: registrska banka



Ponovo slažemo kockice: registrska banka



Par (RADDR, DATA_OUT) nazivamo „pristupom za čitanje“ (eng. *Read Port*), a par (WADDR, DATA_IN) „pristupom za upis“ (eng. *Write Port*)

Analogija sa bankom

Jedan registar u registrskoj banci = jedan sef u trezoru banke



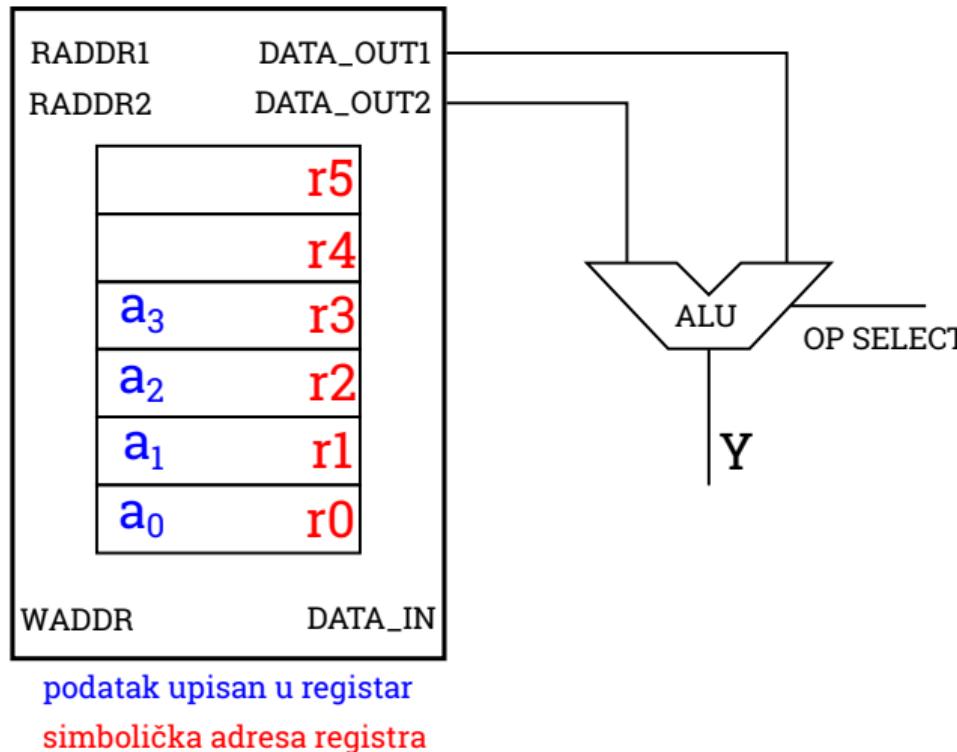
Jedan pristup registrske banke = jedan šalter u banci



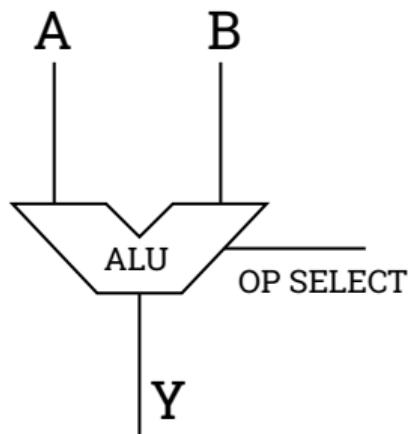
Sefova je mnogo više nego šaltera (registara je mnogo više nego pristupa)

Za razliku od deponovanja i povlačenja predmeta iz sefa, upis u registar briše prethodni sadržaj, dok ga čitanje ne uklanja

Šta je sada potrebno da menjamo tokom računanja?



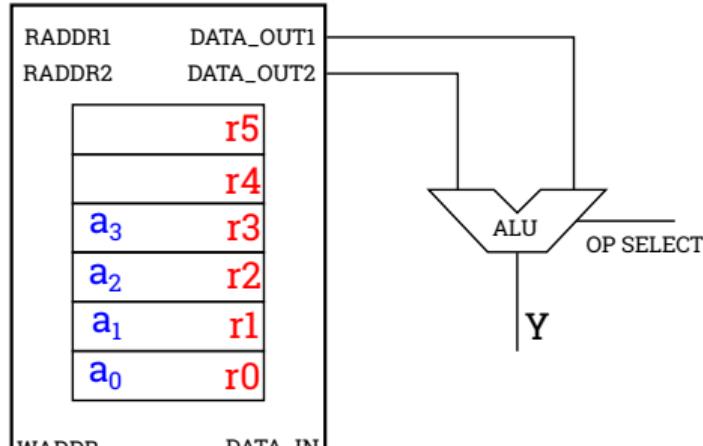
ALU



Potrebno je da menjamo operaciju i operande adrese registara u kojima su smešteni operandi

Koraci izmena

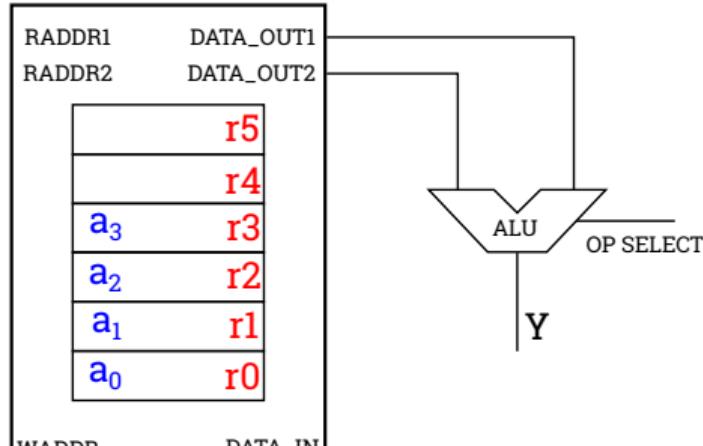
$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



Korak RADDR1 RADDR2 OP
1)

Koraci izmena

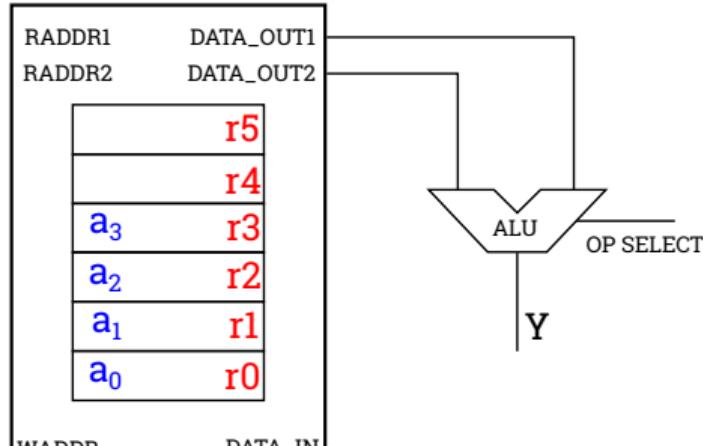
$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



Korak	RADDR1	RADDR2	OP
1)			r0

Koraci izmena

$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$

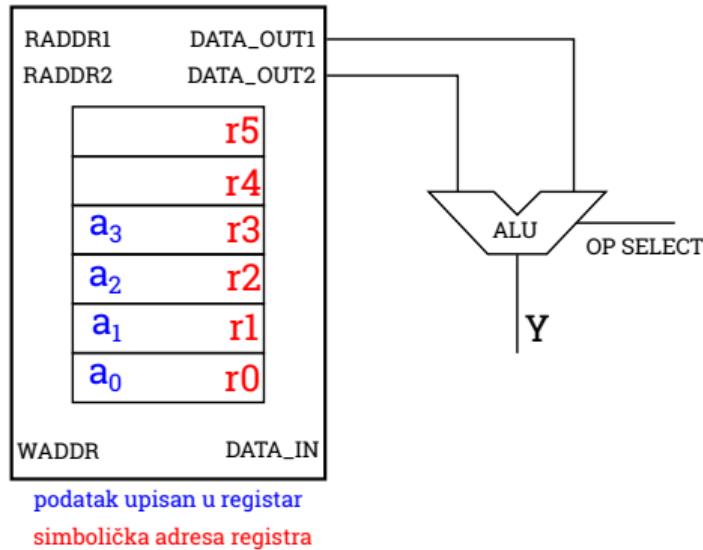


podatak upisan u registar
simbolička adresa registra

Korak	RADDR1	RADDR2	OP
1)	r0	r0	

Koraci izmena

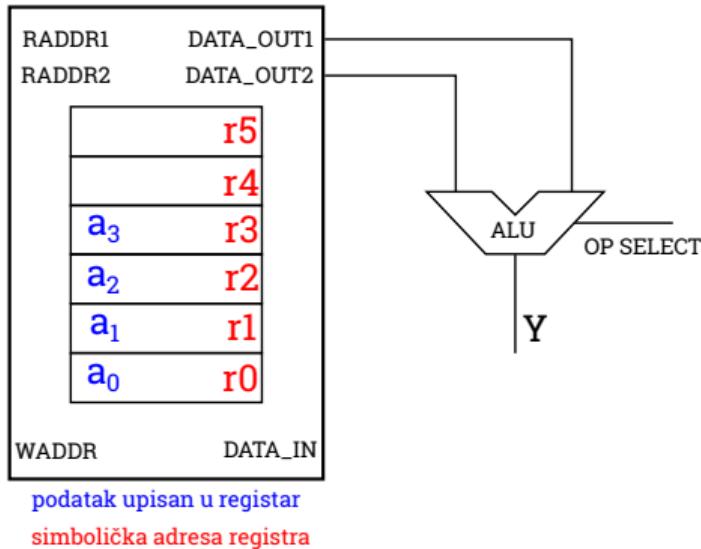
$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



Korak	RADDR1	RADDR2	OP
1)	r0	r0	×

Koraci izmena

$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$

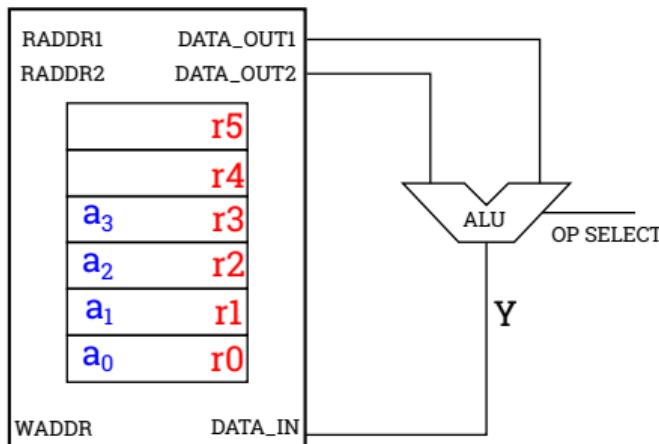


Korak	RADDR1	RADDR2	OP
1)	r0	r0	×

Šta radimo sa rezultatom?

Koraci izmena

$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$

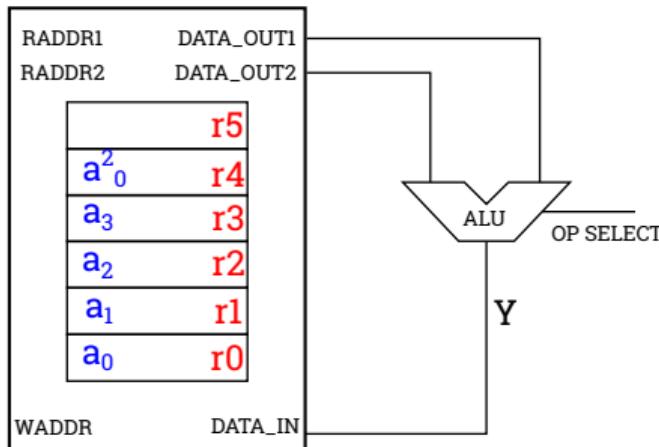


podatak upisan u registar
simbolička adresa registra

Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	

Koraci izmena

$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$

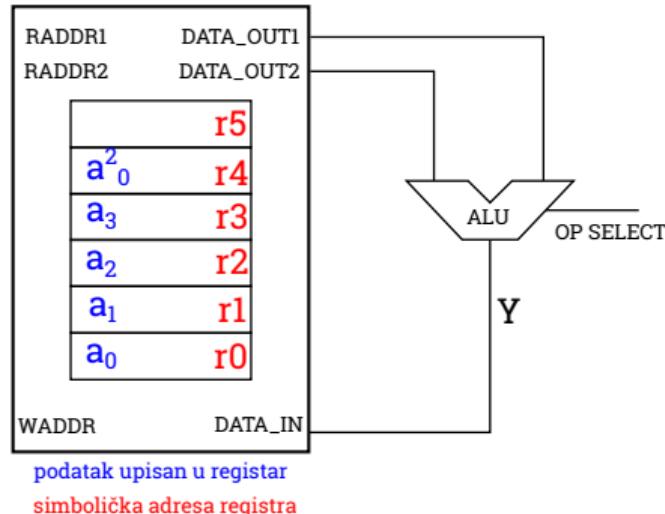


podatak upisan u registar
simbolička adresa registra

Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	\times	r4

Koraci izmena

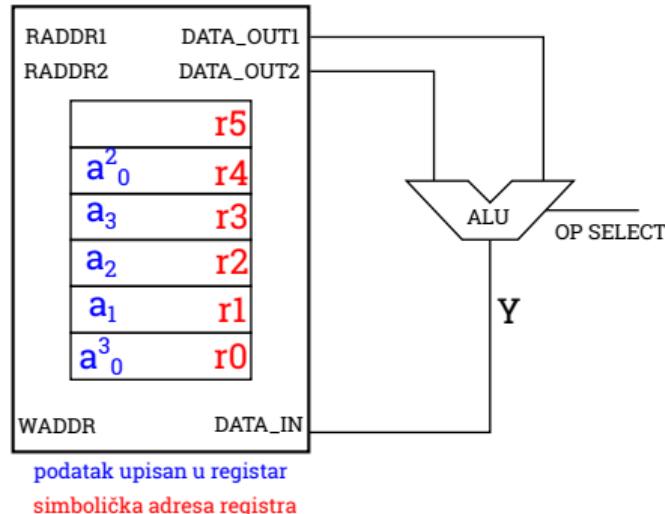
$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	r4
2)	r4	r0	×	r0

Koraci izmena

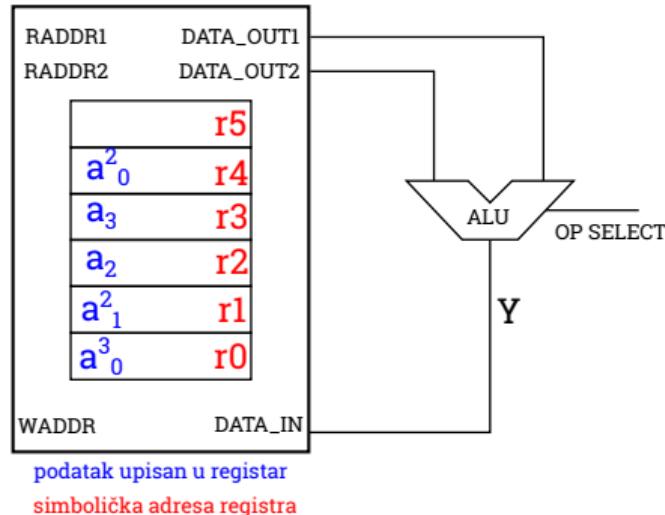
$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	r4
2)	r4	r0	×	r0

Koraci izmena

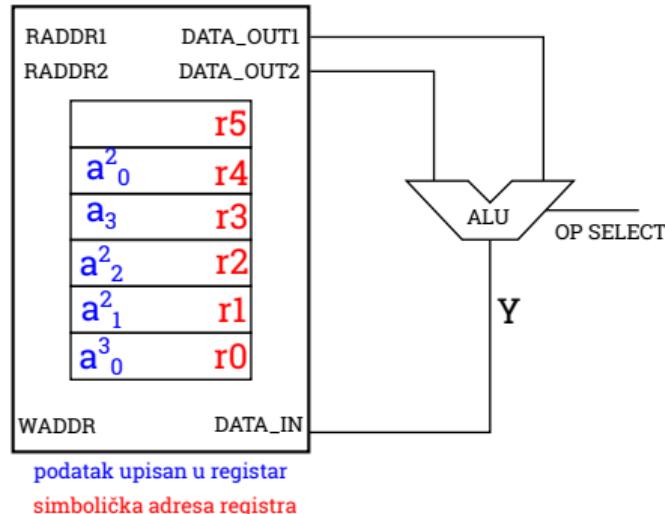
$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	r4
2)	r4	r0	×	r0
3)	r1	r1	×	r1

Koraci izmena

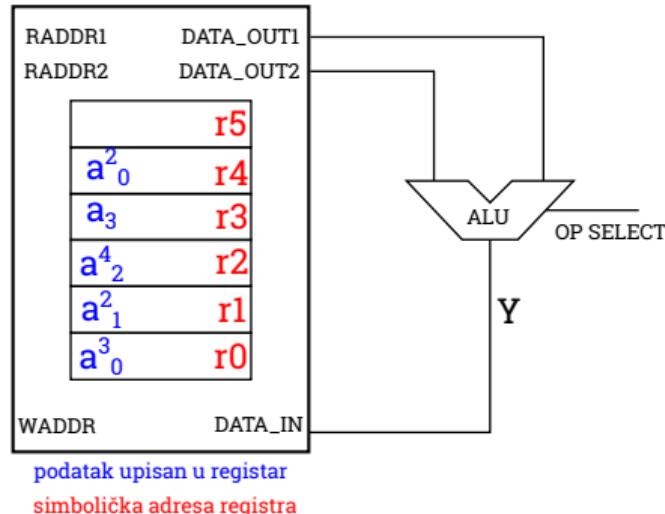
$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	r4
2)	r4	r0	×	r0
3)	r1	r1	×	r1
4)	r2	r2	×	r2

Koraci izmena

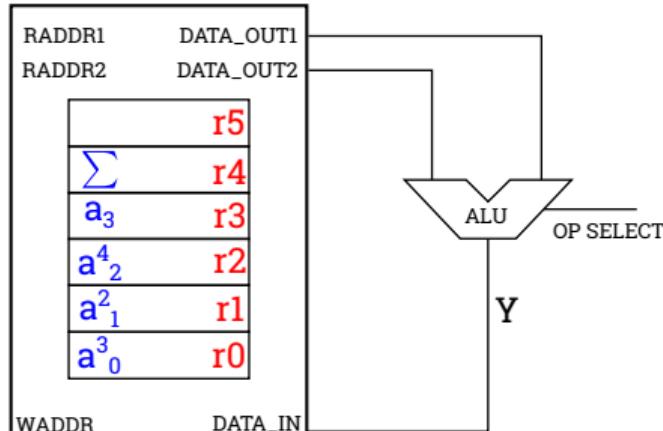
$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	r4
2)	r4	r0	×	r0
3)	r1	r1	×	r1
4)	r2	r2	×	r2
5)	r2	r2	×	r2

Koraci izmena

$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$

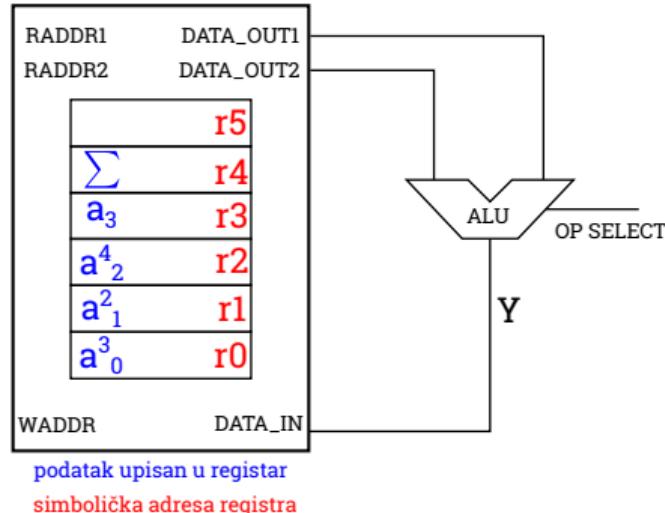


podatak upisan u registar
simbolička adresa registra

Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	r4
2)	r4	r0	×	r0
3)	r1	r1	×	r1
4)	r2	r2	×	r2
5)	r2	r2	×	r2
6)	r0	r1	+	r4

Koraci izmena

$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



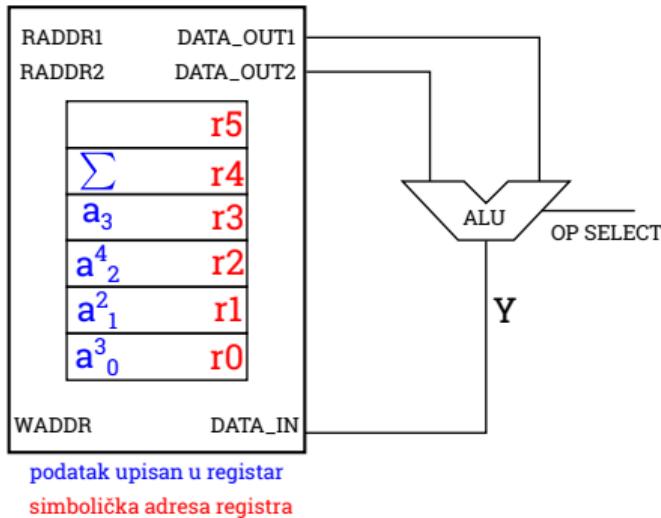
Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	r4
2)	r4	r0	×	r0
3)	r1	r1	×	r1
4)	r2	r2	×	r2
5)	r2	r2	×	r2
6)	r0	r1	+	r4
7)	r4	r2	+	r4
8)	r4	r3	+	r4

Kako da omogućimo računaru da sam izvrši te korake?

Kako da omogućimo računaru da sam izvrši te korake?

Svaka instrukcija je zapravo broj (setite se multipleksera)

$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$

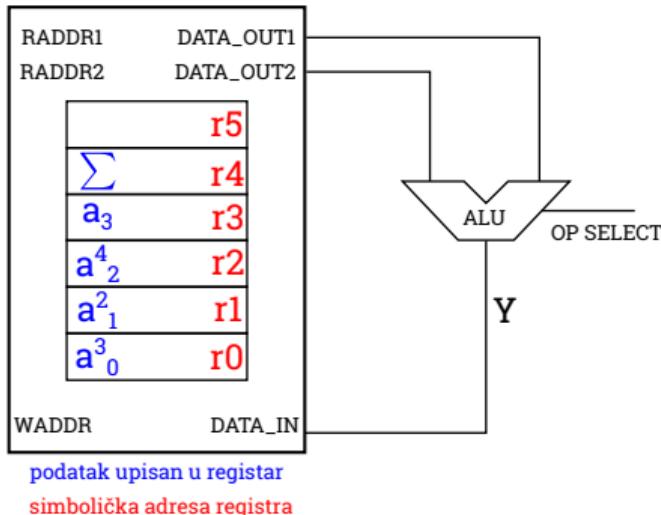


Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	\times	r4
	000001	000001	10	010000
	000001000000110010000			

Kako da omogućimo računaru da sam izvrši te korake?

Svaka instrukcija je zapravo broj (setite se multipleksera)

$$a_0^3 + a_1^2 + a_2^4 + a_3^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



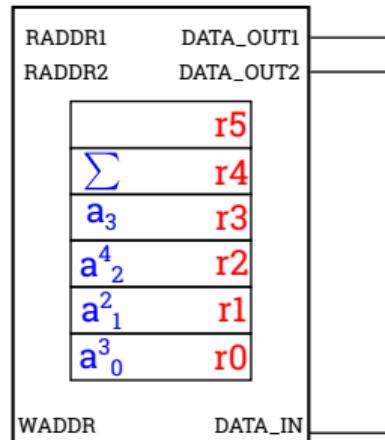
Korak	RADDR1	RADDR2	OP	WADDR
1)	r0	r0	×	r4
	000001	000001	10	010000

00000100000110010000

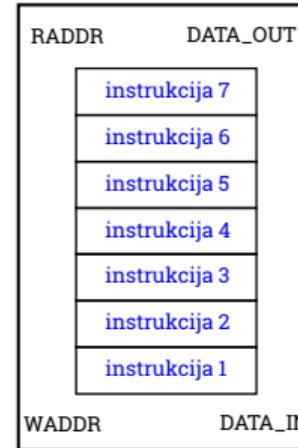
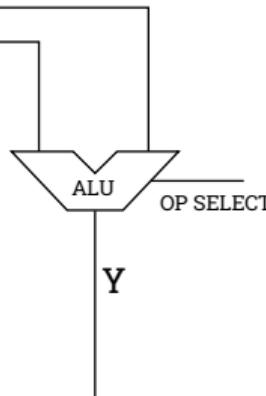
Taj niz brojeva možemo da smestimo u posebnu memoriju

Šta je sada potrebno da menjamo tokom računanja?

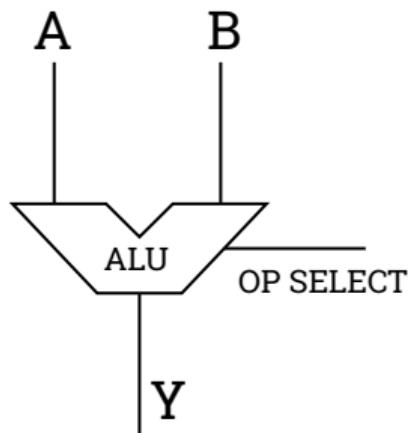
$$a_0^3 + a_1^2 + a_2^4 + a^3 = a_0 \times a_0 \times a_0 + a_1 \times a_1 + a_2 \times a_2 \times a_2 \times a_2 + a_3$$



podatak upisan u registar
simbolička adresa registra



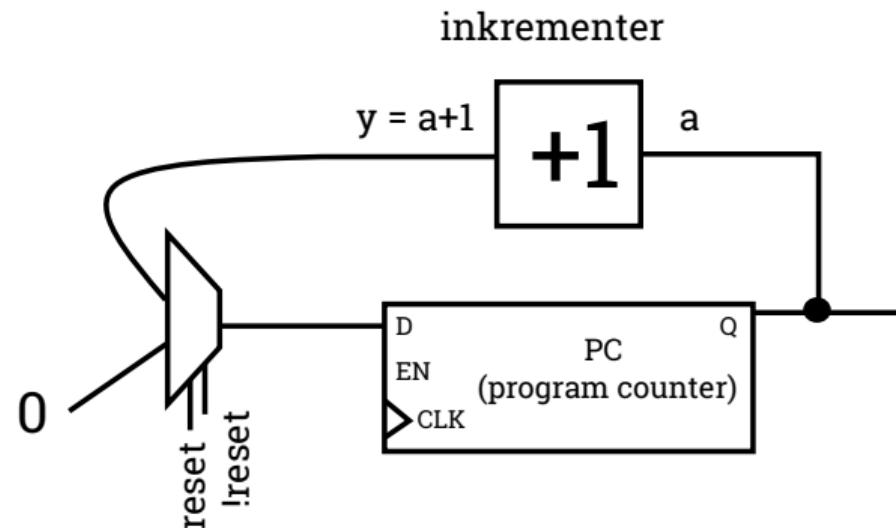
ALU



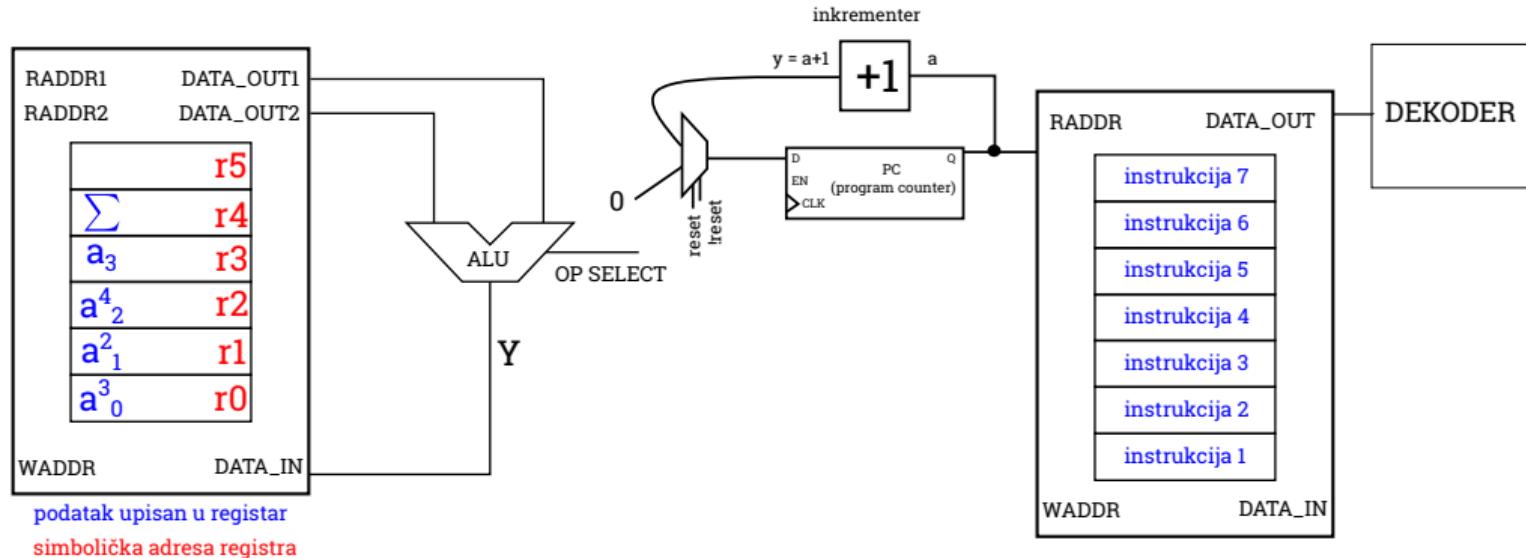
Potrebno je da menjamo operaciju i operande adrese registara u kojima su smešteni operandi adresu instrukcije u memoriji

Kako menjamo adresu?

Ponovo nam je potreban brojač



Ponovo nam je potreban brojač



Ovaj tip računara zovemo „računar sa sačuvanim programom” (eng. *Stored-Program Computer*) jer računar sam čita iz memorije niz instrukcija koje čine program koji je potrebno da izvrši

Taj tip arhitekture računara koji je danas absolutno dominantan nazivamo još i „Fon Nojmanova arhitektura”

Izmislili su je Ekert i Mokli, prilikom dizajna naslednika ENIAC-a, pod imenom EDVAC

Formalni opis koncepata uvedenih u EVAC, prvi je dao Džon fon Nojman (eng. John von Neumann), pa po njemu arhitektura sve do danas nosi ime

Džon fon Nojman



First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

Džon fon Nojman



First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

Šta mislite, šta je po struci bio fon Nojman?

Džon fon Nojman



First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

Tako je, matematičar

Džon fon Nojman



First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

A šta mislite, gde je rođen i gde je završio gimnaziju?





Inač, Jugoslavija je bila peta zemlja u Evropi koja je imala sopstveni digitalni računar

30 НАЈ-ПРЕДМЕТА
30 TOP EXHIBITS

Дигитални диносаурус Digital Dinosaur

У Институту Борис Кидрич – Вилича, 1956. године, под руноводством прва два стручњака из ове области, Тихомира Алексића и Рајка Томовића, започело се са пројектовањем и израдом првог домаћег рачунара, Цифрског електронског рачунара – CER 10. Рађен у комбинованим технологијама електронских цеви, релеја и транзистора, то је био један од првих лет самостално развијених рачунара у Европи. Овај прототип, који је у међувремену пресељен у Институт Михајло Пупин, завршен је и представљен јавности 1960. године, након чега је, уз одређене дораде, инсталiran у СНК (Савезна комисија за нулнеарну енергију), у згради Тонђеве, где га је користило Министарство спољних послова на пословима дешифрована.

До 1975. године прављени су све напреднији модели серије CER рачунара, 11, 20 и 30, 202, 12, 101 и 111, чији корисници су били велика државна предузећа, установе и војска.

CER 10 са својих шест двокрилних металних ормана и периферним јединицама, заузимао је климатизовану просторију од 80 квадратних метара и трошио је 10 kW/h електричне енергије. У њега је било уграђено 1750 електронских цеви и 1500 транзистора. Имао је примарну меморију од 4 килобајта и рачунао је средњом брзином од 50.000 операција у секунди, што је, поређења ради, милион пута мање меморије и 40.000 пута спорије од данашњег проценченог кућног рачунара.

Сачуване делове рачунара CER 10, 12 и 111 Музеју науке и технике поклонио је Институт Михајло Пупин 2006. године.

20

Назив предмета:
Рачунар CER 10 – део
аритметичке јединице
Произвођач:
Институт Михајло Пупин

Inače, Jugoslavija je bila peta zemlja u Evropi koja je imala sopstveni digitalni računar

The screenshot displays a Google Maps interface for a route from Belgrade Centre railway station to the Museum of Science and Technology. The main map shows a complex network of roads and public transport lines. Key landmarks include Barokna kapija, Park Kalemeđan, Muzej Savremene Umetnosti, Brankov Most, Trg Republike, and Novo groblje. The route is color-coded by mode: blue for walking, orange for buses, and light blue for cars. A large inset map provides a detailed view of the walk from the railway station to the museum building, showing the building's exterior and surrounding trees under a clear sky.

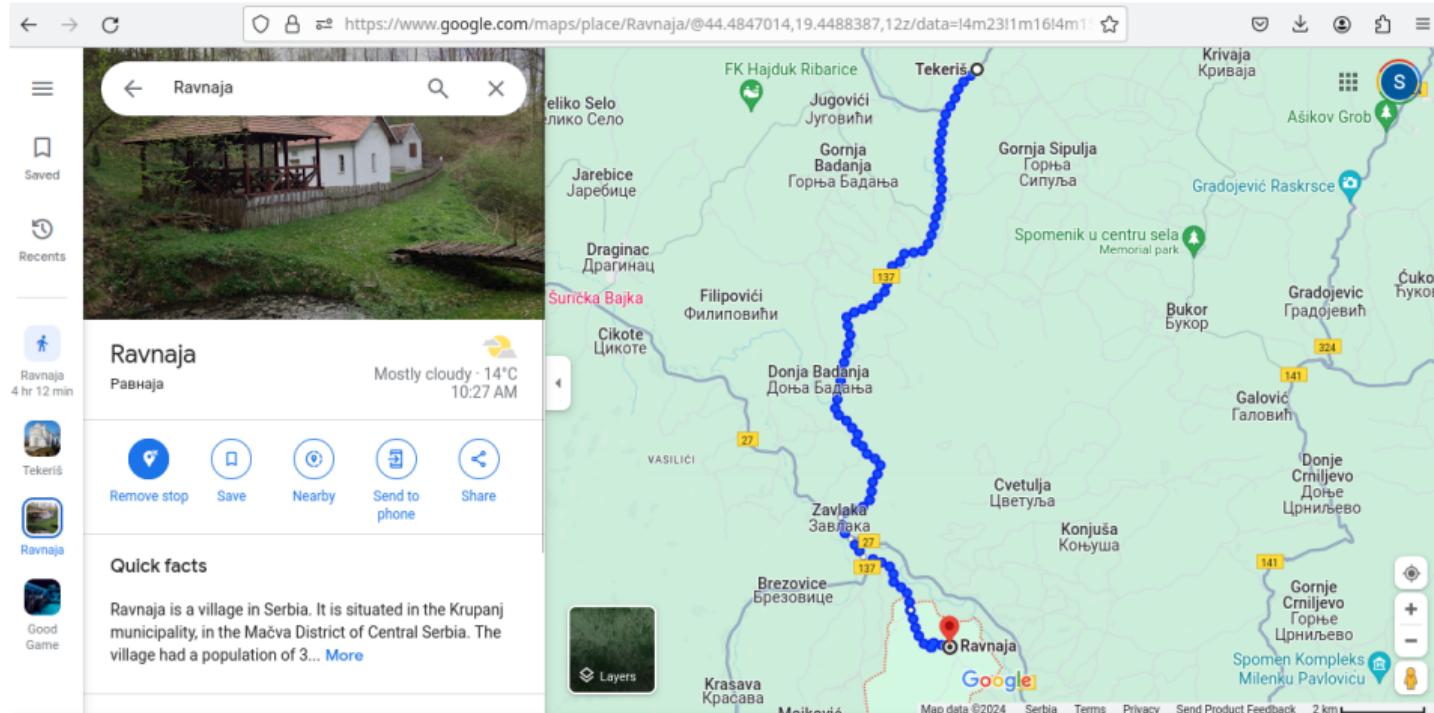
Inače, Jugoslavija je bila peta zemlja u Evropi koja je imala sopstveni digitalni računar



Tihomir Aleksić (Ravnaja, 1922-Beograd, 2004)

Inač, Jugoslavija je bila peta zemlja u Evropi koja je imala sopstveni digitalni računar

CER = Cifarski elektronski računar



Inač, Jugoslavija je bila peta zemlja u Evropi koja je imala sopstveni digitalni računar

The screenshot shows a web browser window with the URL <https://www.sanu.ac.rs/en/academician-rajko-tomovic-a-scientist-of-all-time/>. The page features the official crest of the Serbian Academy of Sciences and Arts, the text "Српска академија наука и уметности", its address "Кнеза Михаила 35, 11000 Београд", phone "+381(0)11 2027-100", and email "sasadir@sanu.ac.rs". It also includes social media links for Instagram and Facebook, and a language switch to English. A large black and white portrait of Academician Rajko Tomović is the central image. The page title is "Academician Rajko Tomović – a scientist of all time". Below the title is a quote from Dejan B. Popović.

Српска академија
наука и уметности

Кнеза Михаила 35
11000 Београд
Телефон: +381(0)11 2027-100

sasadir@sanu.ac.rs

POČETNA
О АКАДЕМИЈИ
ОРГАНИЗАЦИЈА
ОДЕЉЕЊА
ЧЛАНСТВО
БИБЛИОТЕКА
АРХИВ
ГАЛЕРИЈА ЛИКОВНЕ И МУЗИЧКЕ
УМЕТНОСТИ
ГАЛЕРИЈА НАУКЕ И ТЕХНИКЕ
АУДИОВИЗУЕЛНИ АРХИВ
ОГРАНАК У НОВОМ САДУ
ОГРАНАК САНУ У НИШУ
ЦЕНТАР У КРАГУЈЕВЦУ
ИНСТИТУТИ САНУ
ГЕОГРАФСКИ ИНСТИТУТ
"ДОМАЋИ ЈУНІВЕРЗИТЕТ"

80%

SERBIAN ACADEMY
OF SCIENCES AND ARTS

Academician Rajko Tomović – a scientist of all time

Excerpts from the text of Academician Dejan B. Popović, a longtime close associate of Academician Rajko Tomović

Nismo još završili

Naš računar je još uvek veoma ograničen

Za sad smo videli instrukcije tipa

ADD r0, r1, r0

Program koji prevodi ovakve simboličke zapise u odgovarajuće binarne brojeve nazivamo „assemblerom”, a ovakav simbolički jezik „asemblijem” (eng. *assembly*). Za njihov nastanak je zaslužna prof. Ketlin But, koju smo pominjali na prošlom času

Naš računar je još uvek veoma ograničen

Za sad smo videli instrukcije tipa

ADD r0, r1, r0

Prvi problem: oba operanda su specificirana navođenjem imena registra u kom se nalaze. Šta ako je neophodno da baratamo konstantama?

Naš računar je još uvek veoma ograničen

Za sad smo videli instrukcije tipa

ADD r0, r1, r0

Nikakav problem. Uvodimo novu instrukciju tipa

ADDI r0, 0xN, r0

ADDI od eng. *add immediate*, što nalaže dekoderu da umesto da interpretira 0xN kao adresu, direktno prosledi taj broj ALU-u kao konstantan operand

Modovi adresiranja

Razlika imeđu instrukcija ADD i ADDI je u takozvanim „modovima adresiranja”. Ima ih još ali ćemo ih pomenuti tek kada budemo radili nizove u C/C++-u.

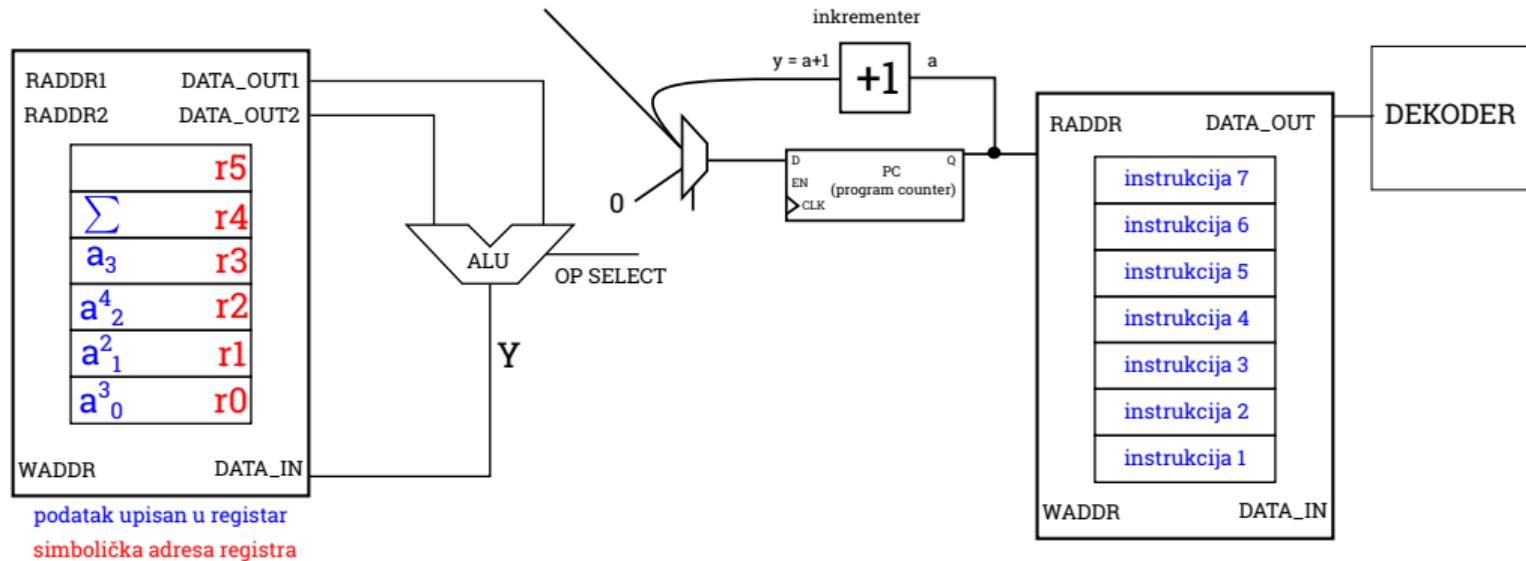
Naš računar je još uvek veoma ograničen

Daleko veći problem je to što može da izvršava samo linearne segmente koda (bez grananja)

No, i to vrlo lako možemo da rešimo

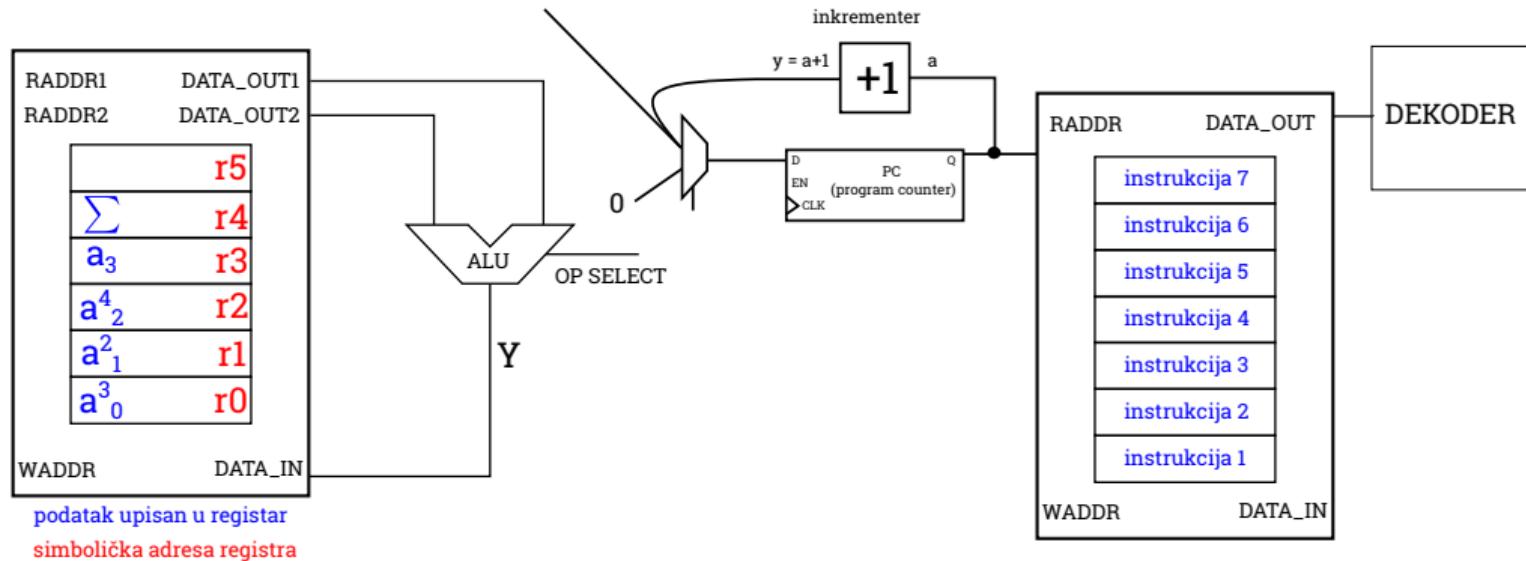
Kako da skočimo na proizvoljnu instrukciju?

Ako upišemo adresu skoka u PC, problem je rešen



Kako da skočimo na proizvoljnu instrukciju?

Ako upišemo adresu skoka u PC, problem je rešen



Gde specificiramo adresu skoka?

Bezuslovni skok

Uvodimo novu instrukciju

JUMP TADDR

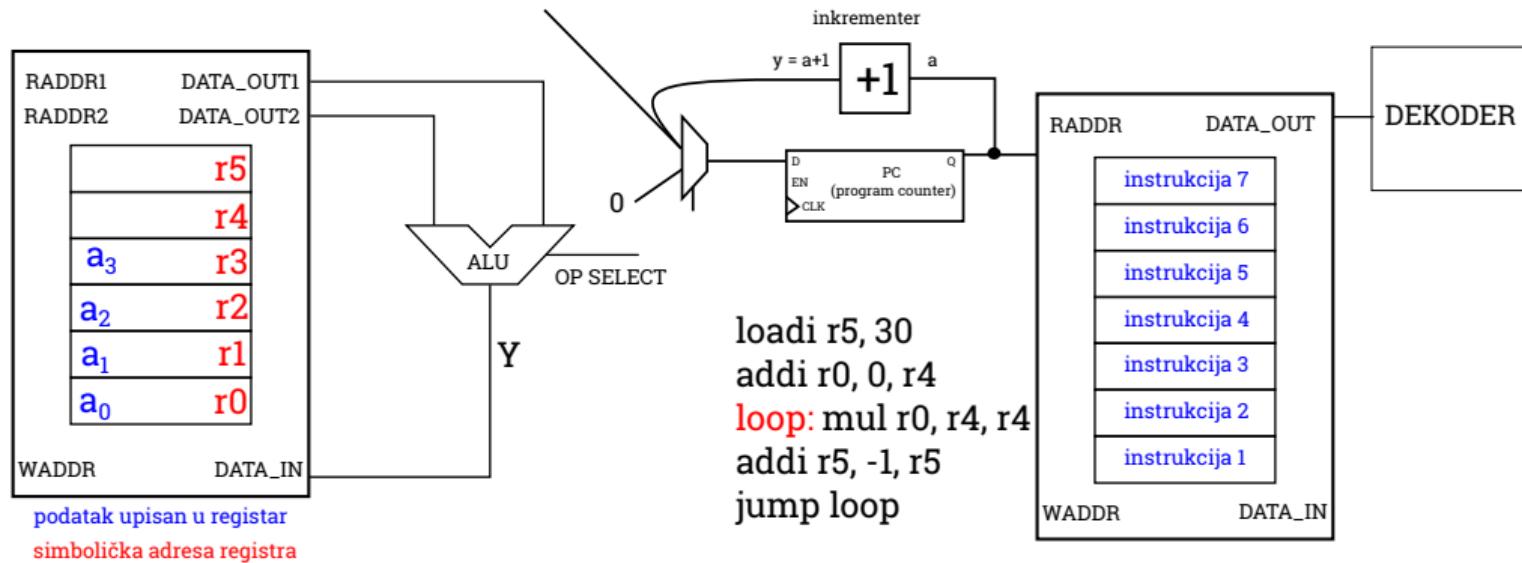
Kada se izvrši, vrednost u PC-u se menja vrednošću konstante TADDR

Kod svih drugih instrukcija koje smo do sada videli, sledeća instrukcija je implicitno na adresi $PC + 1^1$

JUMP eksplisitno navodi adresu sledeće instrukcije

¹Ili $PC + 4$, $PC + 8$, u zavisnosti od toga koliko bita ima procesor i da li je dužina svake instrukcije ista. To su za nas trenutno nebitni detalji, navedeni ovde samo radi kompletnosti.

Šta radi ovaj kod?



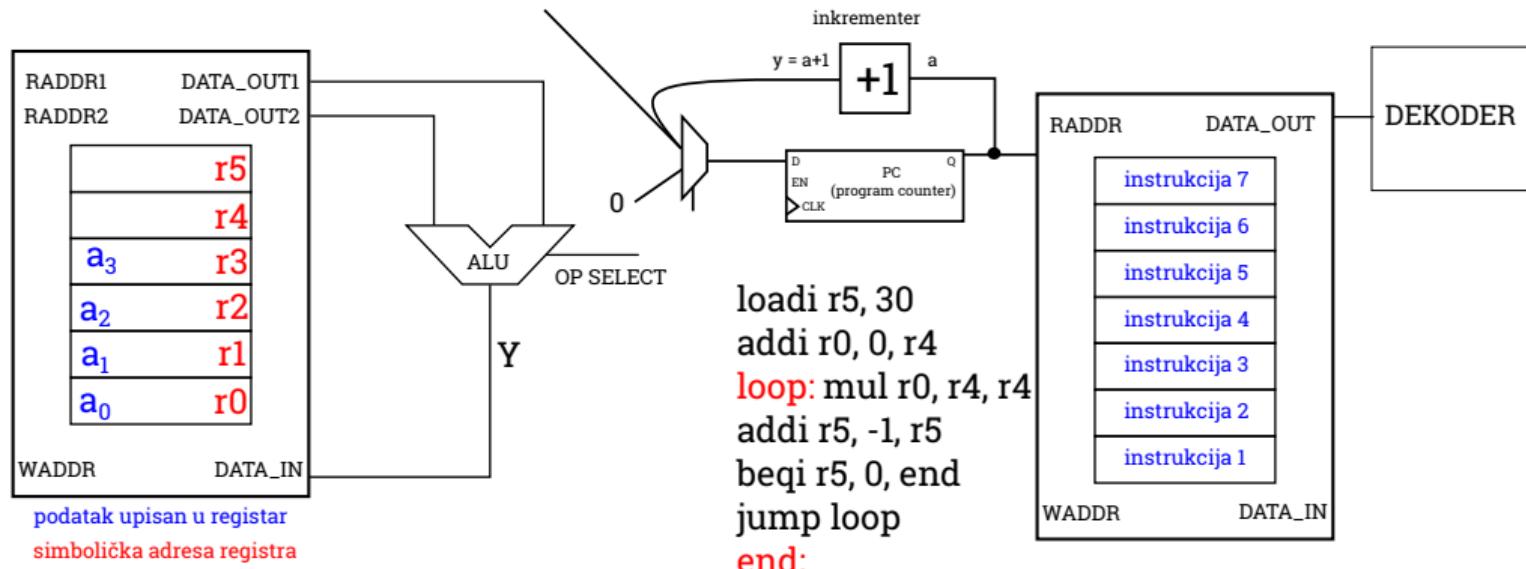
podatak upisan u registar
simbolička adresa registra

Uslovni skokovi

Potrebne su nam i instrukcije skoka koji se izvršava samo ako je neki uslov zadovoljen.

Na primer, BEQI RA, 0xN, TADDR proverava da li register RA sadrži broj 0xN i ako da, učitava TADDR u PC. U suprotnom, ne radi ništa.

Šta radi ovaj kod?

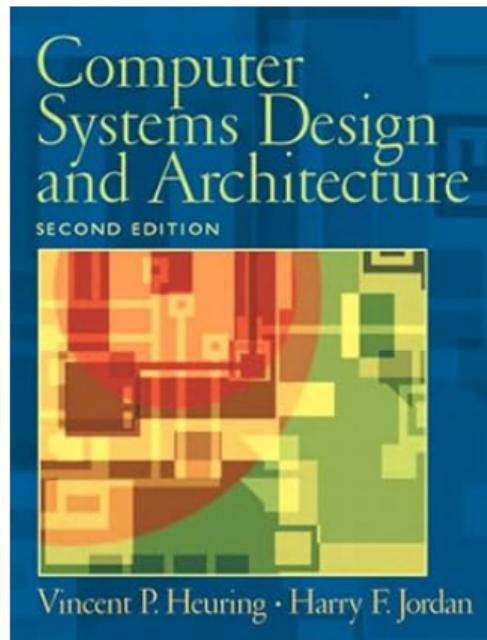


Poziv funkcije je vrlo sličan

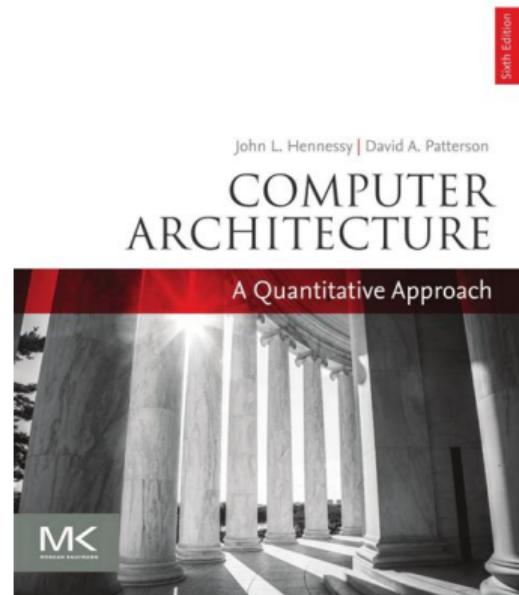
Potrebno je da zapamtimo u nekom rezervnom registru vrednost PC-a. Zatim vršimo bezuslovni skok na adresu na kojoj počinje telo pozvane funkcije. Kada stignemo do kraja, RET instrukcija vraća PC na staru vrednost.

Ima tu još detalja, ali vratićemo se na njih kada za to bude vreme.

Ako neko želi da nauči više o arhitekturi računara



Odlična knjiga za uvod



Nezvanična "Biblija"

Šta radi kompjajler?

Šta je glavni problem asemblira?

Šta je glavni problem asembljija?



Intel® 64 and IA-32 Architectures Software Developer's Manual

Combined Volumes:
1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4

NOTE: This document contains all four volumes of the Intel® 64 and IA-32 Architectures Software Developer's Manual. Basic Architecture Document Number 320650, System Programming Reference A.2, Data Header 425488, System Programming Guide Order Number 321884, Main Specification Order Number 335532. Refer to all four volumes when evaluating your design needs.

IA-32 Instruction Set Quick Reference Card			
<small>See also: Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2, Chapter 1, "Instruction Set Reference," for detailed information on each instruction.</small>			
Op	Op/Reg	Op/Reg/ModR/M	Op/Reg/ModR/M/Disp
MOV			
MOVSB			
MOVSW			
MOVSD			
MOVSL			
MOVSBP			
MOVSWP			
MOVSDP			
MOVSLP			
MOVUPD			
MOVUPW			
MOVUPB			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			
MOVUPLP			
MOVUPDP			
MOVUPLP			
MOVUPBP			
MOVUPWP			

Šta je glavni problem asemblacija?



Intel® 64 and IA-32 Architectures
Software Developer's Manual

**Combined Volumes:
2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4**

NOTE: This document contains all four volumes of the Intel 64 and IA-32 Architectures Software Developer's Manual: Basic Architecture, Order Number 253665; Instruction Set Reference A-2, Order Number 321581; System Programming Guide, Order Number 321584; Model-Specific Registers, Order Number 335552. Refer to all four volumes when evaluating your design needs.

The RISC-V Instruction Set Manual
Volume I: Unprivileged ISA
Document Version 20191213

Editors: Andrew Waterman¹, Krste Asanovic^{1,2}

¹SiFive Inc.,

CS Department, University of C

ive.com, kratzefbe

Program napisan za jedan procesor, ne bi radio na drugom

Šta je glavni problem asemblacija?



Intel® 64 and IA-32 Architectures
Software Developer's Manual

**Combined Volumes:
1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4**

NOTE: This document contains all four volumes of the Intel 64 and IA-32 Architectures Software Developer's Manual: Basic Architecture, Order Number 253665; Instruction Set Reference A-2, Order Number 321581; System Programming Guide, Order Number 321584; Model-Specific Registers, Order Number 335552. Refer to all four volumes when evaluating user design needs.

The RISC-V Instruction Set Manual
Volume I: Unprivileged ISA
Document Version 20191213

Editors: Andrew Waterman¹, Krste Asanovic^{1,2}

¹Silence Inc.

²CS Division, EECS Department, University of California, Berkeley
andrew@aiifive.com, krste@berkeley.edu

No, svaki računar zahteva neke podudarne vrste programa (na primer, operativni sistem)

Ideja

Umesto da pišemo operativni sistem za svaki računar posebno, u odgovarajućem asembliju, napišemo ga u nekom višem jeziku, a proizvođač svakog računara (ili neko drugi) napiše program koji prevodi taj viši izvorni kod u odgovarajući skup instrukcija

Ideja

To stvara ogromnu uštedu programerske radne snage, ali i obezbeđuje kvalitetniji softver (bolja optimizacija, otklanjanje grešaka...)

Programski jezik C je nastao upravo zbog potrebe da se u nekom višem programskom jeziku napiše operativni sistem Unix, kako bi bio portabilan.

Program koji prevodi izvorni kod napisan u višem programskom jeziku u niz instrukcija nazivamo „kompajlerom”

Rekurzivna primena ideje

Pored pukog prevođenja, kompjuter može i značajno da optimizuje kod. Većina raspoloživih optimizacija je opšta i ne zavisi od konkretnog procesora. Stoga se kod prvo prevodi u međureprezentaciju (eng. *intermediate representation (IR)*), koja se zatim optimizuje na različite načine (na primer, redukcijom visine stabla računanja, koju smo već videli, zatim uklanjanjem koda koji se nikada ne izvršava, itd). Ta optimizovana reprezentacija se zatim prevodi u niz instrukcija u fazi „generisanja koda“ (eng. *code generation*). Samo ta poslednja faza zavisi od konkretnog računara za koji generišemo kod.

Nešto o interpretiranim jezicima

Python je interpretiran jezik, što znači da program koji nazivamo interpreterom prevođenje vrši u letu, tokom izvršenja.

Posledice: 1) vreme provedeno u prevođenju ulazi u vreme izvršenja, 2) optimizacije je moguće vršiti samo nad ograničenim segmentima koda, jer interpreter ne sagledava ceo program odjednom. Upravo to je glavni razlog zašto su C/C++ programi i po nekoliko redova veličine puta brži.

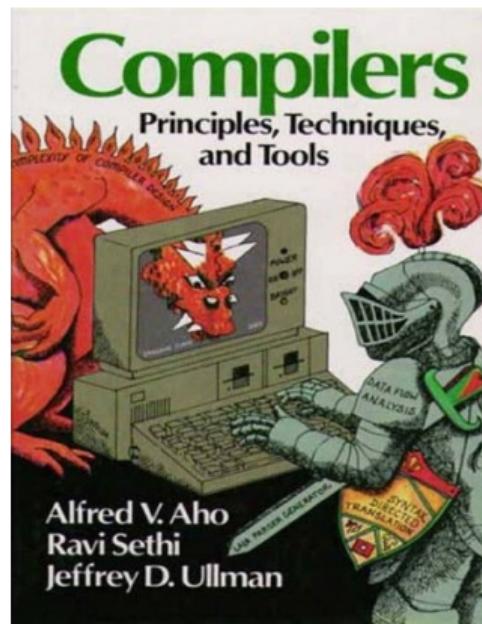
Kako kompjajler implementira stepen sa prošlog časa?

The screenshot shows the Compiler Explorer interface on godbolt.org. On the left, in the C++ source editor (Editor #1), there is a simple function named 'stepen' that calculates the fifth power of an unsigned integer using the `pow` function from `<cmath>`. On the right, in the assembly editor (Editor #1), the generated assembly code for the RISC-V 32-bit target using gcc (trunk) shows the following sequence of instructions:

```
1  stepen(unsigned int):
2      fcvt.d.wu    fa4,a0
3      fmul.d     fa5,fa4,fa4
4      fmul.d     fa5,fa5,fa5
5      fmul.d     fa5,fa5,fa4
6      fcvt.wu.d a0,fa5,rtz
7      ret
```

Na ovom sajtu se možete igrati raznim kompjajlerima i raznim ISA-ma, bez ikakve instalacije

Ako neko želi da nauči više o kompjajlerима



Osnovna struktura C/C++ programa

Zdravo svete

```
//Naš prvi program. Ovo je komentar koji traje do kraja linije.  
#include <iostream>  
/*  
 * Ovo je preprocesorska direktiva koja nalaže preprocesoru  
 * ---programu koji se izvršava neposredno pre kompilacije---  
 * da u kod koji je potrebno kompajlirati uveze sadržinu fajla  
 * iostream u kom su, izmedju ostalih, navedena zaglavljiva  
 * funkcija za ulaz sa tastature i izlaz na ekran.  
 */  
  
int main()  
/*  
 * Definicija glavne funkcije. Naš program se ne izvršava  
 * samostalno na procesoru, već posredstvom operativnog sistema.  
 * Operativni sistem "skace" na početak našeg programa  
 * pozivom ove funkcije.  
 */  
{ //Početak bloka  
  
    std::cout << "Zdravo svete!\n";  
    //Svaka naredba u C/C++-u se završava  
    //takozvanim terminatorom: znakom ;  
  
    //std je takozvani imenski prostor (eng. namespace),  
    //sličan pozivnom broju države. Na primer, +38121 je Novi Sad  
    //a +4121 Lozana.  
  
    return 0; //Ako je program izvršen bez gresaka, po konvenciji,  
    //main vraća operativnom sistemu 0.  
}  
//Kraj bloka
```

Hvala na pažnji