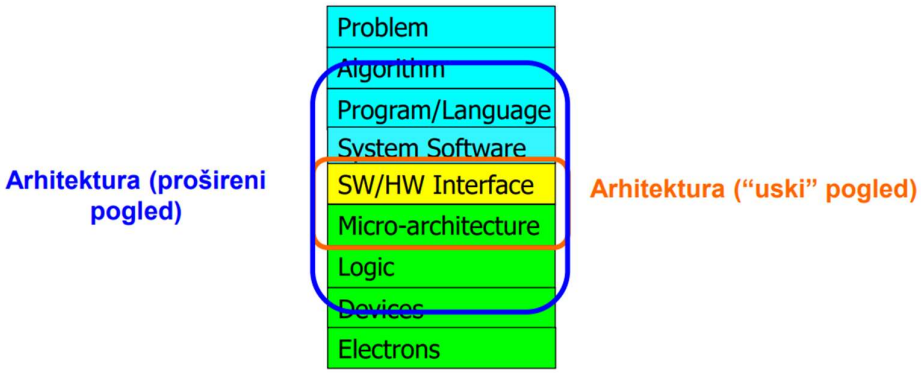
Građa Računala

1. Računalna Arhitektura

- Četiri glavna smjera razvoja računalnih arhitektura:
  - Sigurne/pouzdane arhitekture
  - Energetski efikasne arhitekture
  - Arhitekture sa niskom latencijom, predvidljivi performansi
  - Arhitekture za AI/ML, analizu genoma, medicinu
- Transformacijska hijerarhija

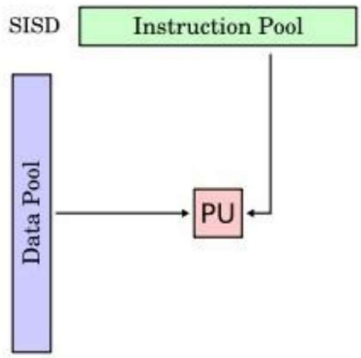


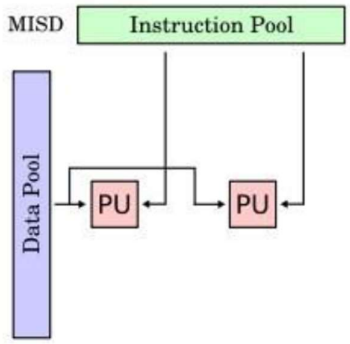
  - Ko-dizajnirati kroz hijerarhiju algoritme prema uređaju
  - Specijalizirati koliko je moguće unutar ciljeva dizajna
- Građa računala/računalna arhitektura
  - Znanost i umjetnost dizajniranja, odabiranja i povezivanja hardverskih komponenti i dizajniranje hardverskih sučelja radi kreiranja računalnog sustava koji je u skladu sa zahtjevima
- Zašto treba upoznati građu računala?
  - Razvoj boljih sustava: bržih, jeftinijih, manjih, pouzdanijih
  - Razvoj boljih aplikacija
  - Razvoj boljih rješenja za probleme
  - Da razumijemo zašto računala rade kako rade

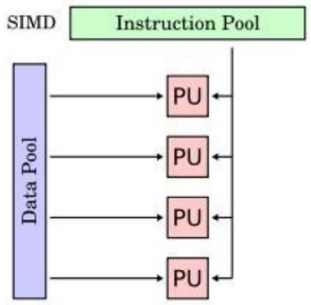
2. Turingov stroj, von Neumannovo računalo, Flynnova klasifikacija

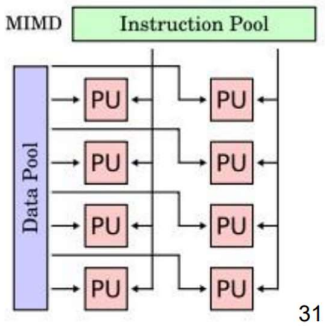
- Obrada podataka
  - Obrada podataka je svrsishodna aktivnost koja ime za cilj dobivanje tražene informacije iz raspoloživih podataka
  - Obrada informacija objedinjuje aktivnosti:
    - Pohrane podataka
    - Obrade velikih količina informacija
    - Slanje, odnosno upućivanje tražene informacije prema odredištu te njezino ponovno pohranjivanje
- U postupku obrade podataka možemo identificirati tri glavne komponente:
  - **Podatci** su objekti u obradi i moraju biti predočeni u obliku koji je prilagođen izvršitelju
  - **Algoritam** predstavlja preciznu uputu ili „recept“ izvršitelju kojom se opisuje transformacija početnih ili ulaznih podataka u procesu obrade u traženu informaciju
  - **Izvršitelj** može biti čovjek ili stroj
- Baviti se samo izvršiteljem – strojem koji se temelji na *računskom modelu* definirao kao **von Neumannov model**
- Računalni modeli se opisuju skupom triju apstrakcija:
  - Temeljnim elementima koji sudjeluju u računanju
  - Modelom kojim se opisuje problem
  - Izvršnim modelom
- Upravljanje slijedom izvršavanja instrukcija može se temeljiti na:
  - Upravljačkom toku (engl. *Control-driven execution*)
  - Toku podataka (engl. *Data-driven*)
  - Upravljanju zahtjevima (engl. *Demand-driven*)
- **Turingov stroj**
  - Turing je uspješno dokazao da nikada ne može postojati nikakva univerzalna algoritamska metoda za određivanje je li prijedlog neodlučiv
  - Računalnoj znanosti je Turingov stroj zanimljiv zato što nam pomaže pri učenju *assembler-a*
- Hijerarhijski model računala
  - Korisnik
  - Korisnički program
  - Preostali dio operacijskog sustava
  - Jezgra operacijskog sustava
  - Sklopovska oprema
- Klasifikacija arhitekture računala
  - Prema načinu izvršavanja instrukcija
    - Arhitektura računala sa upravljačkim tokom (control-flow)
    - Arhitektura računala upravljana tokom podataka (data-flow)
    - Arhitektura računala upravljana zahtjevima (demand driven)

    - **Flynnova klasifikacija**
        - Temelji se na instrukcijskom toku i toku podataka
        - Postoje četiri osnovna tipa arhitekture
            - SISD
            - MISD
            - SIMD
            - MIMD
- SISD (Single Instruction Stream Single Data Stream)
    - Računalo s jednostrukim instrukcijskim tokom i jednostrukim tokom podataka
    - Arhitektura SISD predstavlja arhitekturu sekvencijalnog računala temeljenog na von Neummanovom modelu računala
- MISD (Multiple Instruction Stream Single Dana Stream)
    - Računalo sa višestrukim instrukcijskim tokom i jednostrukim tokom podataka
    - Računala ovog tipa arhitekture ne mogu se fizički realizirati jer je nemoguće ostvariti da se istodobno više različitih instrukcija izvršava na istim podatcima
    - U ovo kategoriju dogovorno uvrštavamo protočna (engl. pipeline) računala
- SIMD (Single Instruction Stream Multiple Data Stream)
    - Računalo sa jednostrukim instrukcijskim tokom i višestrukim tokom podataka
    - U ovu kategoriju se svrstavaju paralelna računala (matrična računala)
    - Obično se sastoje od velikog broja procesora koji istodobno izvršavaju istu instrukciju na različitim podatcima
- MIMD (Multiple Instruction Stream Multiple Data Stream)
    - Računalo s višestrukim instrukcijskim tokom i višestrukim tokom podataka
    - Multi procesorski sustavi
    - Svaki procesor ima pristup zajedničkoj memorijskoj jedinici i svi dijele ulazno izlazne jedinice, a pritom djeluju pod jednim operacijskim sustavom
- **Flynnova klasifikacija slika**









31

- Funkcijske jedinice računala
  - Aritmetička jedinica
  - Upravljačka jedinica
  - Memorija
  - Ulazna jedinica
  - Izlazna jedinica

3. Elementi ALU

- Mikroprocesor ima:
  - Sklopove za rukovanje podatcima
  - Upravljačke sklopove
- Sklopovi za rukovanje podatcima
  - **Aritmetičko – logička jedinica**
  - Akumulator
  - Registri opće namjene
  - Registar uvjeta
  - Adresni registri
  - Segmentni registri
- Uloga ALU
  - Uobičajene operacije koje ALU obavlja:
    - Aritmetičke operacije
    - Logičke operacije
    - Premještanje podataka
  - Ime duguje osnovnim operacijama koje obavlja
  - Sastoji se od različitih vrsta krugova/sklopova
- Komponente dizajna ALU
  - ALU komponente koje rade aritmetičke operacije su dizajnirane oko sklopova koji su dizajnirani za takve operacije
  - ALU komponente koje određuju poslove kao što su FP operacije, ili decimalne operacije su obično puno kompleksnijeg dizajna (*Koprocesor*)
  - Koprocesor radi „u harmoniji“ sa procesorom

- Arhitektura **von Neumannovog računala**
  - Uobičajene gradivne jedinice:
    - ALU
    - Memory
    - Input i Output
    - Kontrolna jedinica
  - ALU mora imati registar koji se naziva *Akumulator*
  - Kontrolna jedinica mora imati registar/brojač koji se zove *PC (Program Counter)*
- Kako radi von Neumannovo računalo?
  - Izvršava ili *emulira* Fetch – Decode – Execute sekvencu
    - **Fetch** – dobivanje instrukcije iz memorije sa adrese koja je zapisana u PC, inkrementima se PC tako da se namjesti na adresu iduće instrukcije
    - **Decode** – dekodiranje instrukcije korištenjem kontrolne jedinice
    - **Execute** – izvršava se program korištenjem podatkovnog toka, kontrolne jedinice i ALU
- Problem dizajna von Neumannovog računala
  - Procesor izvršava jednu po jednu instrukciju (nema paralelizma)
  - Podatci i instrukcije koriste isti tok podataka
  - Memorijski ciklus je dulji od CPU ciklusa
  - Ovo se sve naziva „**usko grlo**“
- Harvard arhitektura – rješava probleme von Neumannovog računala
- Prikaz znakova
  - ASCII – svaki je alfanumerički znak predočen 7-bitnim kodom
  - Unicode
    - Koristi jedinstven broj za svaki znak
    - Njime je opisan svaki znak svih poznatih jezika
    - Koristi 16 bitova
- Prikaz brojeva
  - Binarni sustav
  - Različiti načini prikaza cijelih brojeva:
    - Predznak-apsolutna vrijednost
    - Jedinični ili nepotpuni komplement
    - Potpuni ili dvojni komplement
  - Prikaz brojeve u notaciji potpunim ili dvojim komplementom – najčešće korišteni način za prikaz cijelih brojeva
- Zbrajanje dvaju brojeva
  - Poluzbrajalo; HA
  - Potpuna zbrajalo; FA
  - Paralelno zbrajalo
  - Serijsko zbrajalo
- Oduzimanje dvaju brojeva
  - Poluoduzimalo; HS
  - Potpuno oduzimalo
  - Oduzimanje pomoću komplementa binarnog broja
- **Jednostavan ALU**
  - Više funkcijski digitalni sklop
  - Izvodi aritmetičke i logičke operacije

    - Osnovana građevna sastavnica jednog stupnja ALU je **potpuno zbrajalo**
    - Povezivanje n stupnjeva ALU dobije se **paralelno zbrajalo**
    - Jednostavan ALU b trebao moći izvesti osam operacija; da bi se ovo desio potrebni su određeni logički sklopovi na ulazima
- Logička sekcija
    - Treba podržavati četiri osnovne operacije **I, ILI, NE** i **ISKLJUČIVO ILI**

4. Osnovni dijelovi računala

- Podsustavi računala
    - Kućište sa napajanjem
    - Matična ploča
    - Procesor
    - Memorija
    - Diskovni podsustav
    - Video podsustav
    - Audio podsustav
    - Mrežni podsustav
    - Ulazno-izlazni uređaji
    - Ostalo
- **Čipset** (*Chipset*)
    - Sklopovi kojima se ostvaruju sučelja između pojedinih sabirničkih struktura u osobnim računalima
    - Čipset se sastoji od dva glavna čipa:
        - Sjeverni čip (*NorthBridge*) – brze sabirnice
        - Južni čip (*SouthBridge*) – sve spore sabirnice

5. Elementi CU

- **Von Neumannovo računalo**
    - „Stored Program Concept“ – instrukcije su pohranjene u memoriji skupa sa podatcima u formatu koji je čitljiv CPU-u
    - Instrukcije se izvode serijski (sekvencijalno) prateći kontrolni tok programa
- Što sve radi CU?
    - Generira upravljačke signale
    - Koordinira sve aktivnosti unutar mikroprocesora
    - Sinkronizira prijenos podataka i komunikaciju modela
    - Pribavlja, dekodira i omogućuje izvođenje instrukcija
    - Komunicira s ostalim komponentama mikroračunala preko sabirnica
    - Upravlja odgovorima na vanjske signale
- Kako radi kontrolna jedinica?
    - Prima ulazne informacije koje pretvara u kontrolne signale
    - Kontrolni signali se šalju procesoru
    - Procesor govori dostupnim uređajima koje operacije treba izvršiti

- Jedinice koje sudjeluju u izvršavanju
    - ALU
    - Programski brojač
    - Memorija
    - CPU registar
    - Interni registar, multiplekseri…
    - Interna sabirnica
    - Kontrolna jedinica
- Što je zapravo kontrolna jedinica?
    - Jedinica stanja koja generira kontrolna stanja ovisno o ulazu, dok se izlaz generira s obzirom na uvjete i podatke na ulazu
    - Bitni parametri kontrolne jedinice:
        - Processor Clock
        - Instrukcijski registar (IR)
        - Zastavice, kontrolni kodovi
        - Eksterna kontrolna sabirnica
- Tipovi instrukcija
    - Generalne instrukcije – izvršavaju se direktno na procesoru
    - Instrukcije sa grananjem – implementirane direktno od strane kontrolne jedinice
- Vrste kontrolnih jedinica
    - *Hardwired* logika CU
        - Redoslijed operacija ovisi o tome kako su logički elementi spojeni
    - *Micro-programmed* logika CU
        - Sadržaj memorijske jedinice nakon dekodiranja postaje baza za generiranje kontrolnih signala za trenutno stanje izvršavanja
        - Vrste
            - Sa pohranom u jednoj razini
            - Sa pohranom u dvije razine – memorija za instrukcije i memorija za nanoinstrukcij

6. Sigurnost hardvera

- Memorija je usko grlo
- **RowHammer** je primjenjiv na sve vrste memorije, pogotovo DDR3+ memorije
- Što je novija tehnologija, situacija je lošija
- **Row hammer** (also written as **rowhammer**) is a computer security exploit that takes advantage of an unintended and undesirable side effect in dynamic random-access memory (DRAM) in which memory cells interact electrically between themselves by leaking their charges, possibly changing the contents of nearby memory rows that were not addressed in the original memory access.
- **Meltdown** i **Specter**
- Iskorištavaju spekulativno istraživanje
    - Izvršava se dio koda prije nego što znamo da li je to potrebno
        - Procesor ovo radi kako bi štedio na vremenu
    - Spekulativno istraživanje ostavlja tragove tajnih podataka u cache memoriji procesora

- Maliciozni program može provjeravati cache memorije da bi došao do zaključka gdje se nalaze tajni podatci kojima ne bi smio moći pristupiti
- Maliciozni program također može prisiliti drugi program da spekulativno izvrši kod koji ostavlja tragove (tajne podatke)

7. Asmebler zadatci

- Fill 512 bytes of memory from location $200 with value $1
  - LDA #$01
    LDX #$00
    loop:
    STA $200, X
    STA $300, X
    INX
    BNE loop
- Fill half of the memory with one and other half with another colour
  - LDX #$00
    loop:
    LDA #$01
    STA $200, X
    STA $300, X
    LDA #$02
    STA $400, X
    STA $500, X
    INX
    BNE loop
- Store one 32 bit number in memory starting at $200, another at $210. Choose how to store the number yourselve. Add those numbers and store result in $220.
  - LDA #$01
    STA $200
    STA $201
    STA $202
    STA $203
    LDA #$01
    STA $210
    STA $211
    STA $212
    STA $213
    LDX #$03
    loop1:
    LDA $200, X
    ADC $210, X
    STA $220, X
    DEX
    BNE loop1
    LDA $200
    ADC $210

STA $220

- Create two dots in the middle of the „video memory”. Animate them moving in oposite directions, first on X then on Y axis. Repeat until reset.
    - LDA #$01
      LDY #$00
      loop1:
      LDA #$00
      STA $42E, X
      STA $430, Y
      LDA #$01
      INX
      DEY
      STA $42E, X
      STA $430, Y
      BNE loop1
- https://www.masswerk.at/6502/6502_instruction_set.html - assembler komande objašnjne i prikazane

8. Primjer ispita

- Kod dizajna ALU što se oblikuje prvo: aritmetički ili logički dio (1 bod) i zašto (2 boda)?
    - Dio koji se oblikuje prvi varira ovisno o potrebama i namjeni specifičnog ALU-a. Najčešće su oblikovani i povezani zajedno, ali u nekim situacijama iz bilo kojeg razloga, bilo koji od njih može biti oblikovan prije drugog.
- Zapišite brojku 3FFA u little endian (1 bod) i u big endian (1 bod) zapisu.
    - Little endian: AFF3
    - Big endian: 3FFA
- Demonstrirajte 8 bitno zbrajanje bez prijenosa.
    - Logički sklopovi (ja mislim ali ne znam)
- Imamo dvije hipotetske ALU, zadane tablicama stanja. Jedinica A ima 16 mogućih operacija. Jedinica B ima 8. Ako pretpostavimo da je logički dio obje jedinice identičan koja od njih će nužno imati više sklopova i osnovnih logičkih elemenata?
    - Jedinica A će imati više sklopova jer može raditi 16 mogućih operacija. Što više operacija to je generalno kompliciranija shema i zahtjeva više sklopova i osnovnih logičkih elemenata.
- Demonstrirajte overflow s predznakom (signed overflow).
    - Ako u 8 bitnom sustavu koji koristi predznake postoje vrijednosti koje variraju između -128 i 127. Ako zbrojimo dva pozitivna broja čiji rezultat prelazi vrijednost 127 desiti će se overflow.
    - Mogući primjer; izbacilo je u akumulatoru da ima overflow-a:
      LDX #$01
      LDA #$80
      STA $0400
      SBC $0400,x
- Objasnite na primjerima koje su uloge kontrolne jedinice (CU)?
    - Generira upravljačke signale (Kada se nešto treba desiti, CU generira signal koji obavještava ostale dijelove što i tko treba raditi)

    - Koordinira sve aktivnosti unutar mikroprocesora (Određuje šta se i kad dešava. Ako se nešto izvršava unutar procesora, CU je odgovora za kako i kojim redoslijedom se dešava)
    - Sinkronizira prijenos podataka i komunikaciju modela (Za sve podatke koje CU fetch-a iz memorije on određuje, na temelju zastavica (program counter-a), kojim redoslijedom će se izvršiti)
    - Pribavlja, dekodira i omogućuje izvođenje instrukcija (Ako mi napišemo kod u program i želimo da se on izvrši. Kada kliknemo da se izvrši naš CU prima nekakve komande, nekim redoslijedom, dekodira ih i određuje što se dalje treba desiti)
    - Komunicira s ostalim komponentama mikroračunala preko sabirnica(Šalje signale kada netko treba nešto napraviti. Npr. kad uzima stvari iz memorije i kada te stvari šalje drugim komponentama da ih obrade)
    - Upravlja odgovorima na vanjske signale (Ako mi pošaljemo dorađene stvari na ulaz CU-a dobiti ćemo i neki odgovor koji je on odlučio. Kao npr. riješen kod ako smo poslali neke instrukcije na ulaze)
- Što se kolokvijalno naziva FE (Front End) (1 bod), a što BE (Back End) (1 bod) kod procesora i koje zadaće izvršavaju (2 boda)?
    - Front end je naziv za dio procesora koji fetch-a, opisuje i priprema instrukcije, a back end je dio procesora kojih ih obradi, mapira i daje odgovor na temelju obrade. Ovo su dvije vrste dizajniranja procesora.
- Korištenjem 6502 asemblera popunite svaku parnu lokaciju između $200 i $2FF sa vrijednosti decimalno 13.
    - LDA #$15
      LDX #$00
      Loop:
      STA $200,x
      INX
      INX
      BNE Loop
- Prethodni zadatak proširite tako da popunjava i lokacije između $300 i $3FF ali popunite neparne lokacije.
    - LDA #$15
      LDX #$FF
      Loopi:
      STA $300,x
      INX
      INX
      BNE Loopi
- Što je vremenski brže - pristup registru ili pristup memoriji (0.5 bodova) i zašto (0.5 bodova)?
    - Pristup registru je brži jer se nalazi direktno na CPU.
- Kako branch predictor (sklop koji služi za predviđanje grananja) pomaže brzini izvršavanja instrukcija?
    - Branch predictor pomaže brzini tako što pogađa što će biti rezultat neke Branch komande prije nego što se ona desi. Te ako dobro pogodi ubrzava rad procesora. Procesor unaprijed radi i pogađa rezultate Branch instrukcija te tako već,

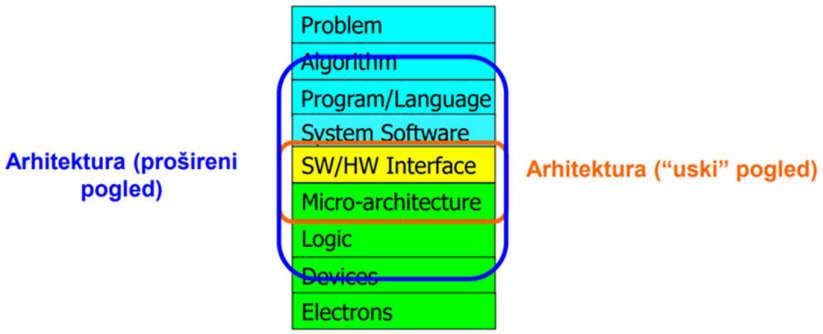
teoretski, zna rezultat i prije nego ta instrukcija dođe na red, pa kad dođe na red izvrši se puno brže nego da je u tom trenutku tek krenuo obrađivati tu instrukciju.

- Izračunajte prostornu (1 bod) i vremensku (1 bod) složenost oba rješenja iz prethodnih zadataka (količinu zauzete memorije i broj ciklusa za cijelo izvršavanje).
    - IDK man...
- Na lokacijama $400 i $410 nalaze se tri bajta koja sadržavaju redom: praznu lokaciju i 16 bitni broj u little endian zapisu. Izračunajte zbroj ova dva broja i zapišite ga na lokaciju $420 koristeći registar Y za spremanje potencijalnog prijenosa (carry)
    - LDA #$FF
      STA $0401
      LDA #$FF
      STA $0402
      LDA #$FF
      STA $0411
      LDA #$FF
      STA $0412
      LDA #$00
      CLC
      LDA $402
      ADC $412
      STA $420
      ADC $401
      ADC $411
      STA $421
      LDA $400
      ADC $410
      STA $422
- Kakve sve vrste instrukcija poznaje kontrolna jedinica (1 bod) i po čemu se te vrste instrukcija razlikuju (2 boda)?
    - Generalne instrukcije – izvršavaju se direktno na procesoru, sekvencijalno kako dolaze na izvršavanje
    - Instrukcije sa grananjem (branching instructions) – implementirane direktno od strane kontrolne jedinice
    - Razlikuju se po tome kako ih CU izvršava. Generalne instrukcije idu sekvencijalno kako dođu, a instrukcije sa grananjem koriste PC kako bi se mogle vratiti na originalnu lokaciju kada obave grananje.
- Predložite - ma koliko potencijalno bili nemogući - barem dva, što jednostavnija načina za rješavanje RowHammer problema.
    - Povećaj broj refresha na suludo velike količine
    - ECC memorija; Memorija ima ekstra bitove koji joj omogućuju da popravi greške u jednom bitu. U idealnom svijetu imamo dovoljno doadtnih bitova da popravimo svaki bit originalne informacije
- Koja je razlika između ranjivosti Spectre i RowHammer?
    - Specter koristi spekulativno adresiranje CPU-a, kako bi probio sustav, tjera CPU da radi spekulativno adresiranje kad to nije potrebno, a RowHammer koristi sam dizajn DRAM-a kako bi promijenio bitove u informaciji.

- Zadatak 1 rješite korištenjem subroutine, lokaciju prvog broja rutina mora pročitati sa $10, $11, drugi i treći broj se nalazi na lokacijama +$10 i +$20. Primjerice ako je na lokaciji $10 zapisano 00 a na lokaciji $11 vrijednost $03, prvo broj koji rutina mora zbrojiti nalazi se stvarno na lokaciji $300, drugi na $310 a rezultat na $320.
    - A
- Koje centralne procesne jedinice (CPU) ne "boluju" od Meltdown ranjivosti (1 bod) i zašto (1bod)?
    - Samo Intel procesori boluju od Meltdown-a, a procesori drugih kompanija kao AMD ili ARM ne jer koriste drugačiju mikrarhitekturu koja nema taj specifični problem, ali i dalje mogu patiti od sličnih explot-a.

9. Primjer ispita 2

- Why do we consider circuits as black boxes most of the time?
    - Unutrašnjost je kompleksna te je nama samo bitno da znamo da kad na ulaze dođemo određene input-e na izlazu ćemo dobiti određene output-e.
- How does signed adding of numbers work? Show a simple example
    - Overflow sa predznakom iz prijašnje ispita; I dalje ne znam primjer, ali mislim da se svodi na to da ako zbrajaš dva 8-bitna broja da ako u rezultatu dobiješ deveti bit zbog carry-a da će sustav ako nije kalibriran misliti da je rezultat negativan.
    - Mogući primjer; izbacilo je u akumulatoru da ima overflow-a:
    - LDX #$01
      LDA #$80
      STA $0400
      SBC $0400,x
- What are basic parts of ALU?
    - Arithmetic Unit
    - Logic Unit
    - Registers
    - Control Unit
    - Flags Register
    - Multiplexer (MUX)
    - Decoder
- Explain the computer architecture transformation architecture – computer architecture narrow (2 points) and expanded view (2 points).



- Based on a set of design goals, how would you synthetize an ALU in accordance to the design specification (2 points)? Feel free to use an example to explain the concept (1 point).
    - Prvo bih pogledao što moj ALU mora raditi, točnije koje logičke i aritmetičke operacije mora podržavati. Te bih na temelju toga istovremeno sintetizirao logičku i aritmetički jedinicu. Npr. ako jedna od glavnih operacija koje moji ALU treba raditi su zbrajanje i oduzimanje. Ne ću raditi dva zasebna sklopa od kojih jedan zbraja a drugi oduzima, nego ću rađe napraviti jedan sklop koji može normalno zbrajati, a oduzimanje radi preko komplementa.
- What is the behaviour of N, V i C flags when we subtract $1 from $80?
    - N flag ostaje na 0, V i C flag se mjenaju u 1. Ovo nam govori da vrijednost nije negativna, ali da postoji overflow i carry.
- Memory locations starting at $400 and $410 contain three bytes, in order: empty byte, 16 bit number in big endian notation. Add those numbers and store the result to location starting at $420 and use the empty byte to store the carry bit.
    - LDA #$FF
      STA $0401
      LDA #$FF
      STA $0402
      LDA #$FF
      STA $0411
      LDA #$FF
      STA $0412
      LDA #$00
      CLC
      LDA $402
      ADC $412
      STA $422
      ADC $401
      ADC $411
      STA $421
      LDA $400
      ADC $410
      STA $420
- Explain Meltdown vulnerability of CPU and its subcomponents and how it influences security.
    - Meltdown je primarno problem za CPU-ove koji rade spekulativno izvođenje. Dopušta napadačima da pristupe sadržaju memorije dok CPU izvodi spekulativno izvođenje. Utječe na securitiy tako što tjera proizvođače hardvera i softvera da naknadno i konzistentno popravljaju i update-aju svoje sustave i hardver.
- Explain how superscalar CPU's work (2 points). What are architectural requirements to create a superscalar CPU (2 points)?
    - CPU-ovi dizajnirani da vrše više instrukcija istovremeno, koristeći paralelizam kako bi poboljšali performanse.
    - Multiple Instruction Multiple Dana

- Solve the task from the previous question using a subroutine (2 points). Enable the user to pass location of the numbers and the result using locations $10, $20 and $30 (1 point)
    - AA
- Imagine that you're working in a Fab that produces computer chips. You're the head of production of the facility, and you're responsible towards your clients (chip designer companies, for example - NVIDIA) to implement security solutions so that even the vetted (security checked) Fab's staff can't introduce any type of hardware changes to client's chip design. What would you do to make sure that – for example – hardware trojans are never introduced to the chip production phase in your Fab?
    - Not a fucking clue
    - Word, my ni**a, word
- Using assembly language on the 6502 fill every third location from $200 to $2FF starting with the first one with decimal value of 15.
    - LDA #$15
      LDX #$Ff
      Loop:
      STA $1FD,x
      DEX
      DEX
      DEX
      BNE Loop
      STA $2FF
- Modify previous example to include memory from $300 to $3FF but first location should be blank.
    - LDA #$15
      LDX #$Ff
      Loop:
      STA $1FF,x
      STA $2FE,x
      DEX
      DEX
      DEX
      BNE Loop
- Which three stages (plus the potential forth stage) does the instruction go through as a part of its execution cycle?
    - Učita vrijednost u X i A, sprema vrijednost u A na određenu lokaciju i deinkrementira X tri puta.
- What is the size and number of cycles taken by tasks in the previous practical questions?
    - IDK
- How does out-of-order execution (1 point) and speculative execution (2 points) influence actual application execution on a CPU?
    - Generalno ubrza eksekuciju ali donosi i određene sigurnosne rizike.