Computer architecure

6502 assembly

Instruction set

- Addressing modes
  - Immediate
  - Absolute
  - Absolute indexed
  - Indirect indexed
- Instruction categories
  - Memory load and store
  - R2R transfer
  - Stack
  - Arthmetic
  - Logical
  - Branching
  - Call and return
  - Processor flag
  - Interupt related
  - Noop

Addresing

- Immediate
- Absolute
- Absolute indexed
- Indirect indexed

Immediate addressing

```
LDA #$04
CLC
ADC #$03
ADC #$02
ADC #$01
```

Absolute addressing

```
LDA #$04
STA $0203
LDA #$03
STA $0202
LDA #$02
STA $0201
LDA #$01
STA $0200

; Add four bytes together using absolute addressing mode
LDA $0203
CLC
ADC $0202
ADC $0201
ADC $0200
```

Absolute indexed addressing

- ```
  LDX #$03
  CLC
  LDA $0200, X
  DEX
  ADC $0200, X
  DEX
  ADC $0200, X
  DEX
  ADC $0200, X
  ```
  ```

Looping it

```
LDX #$03
LDA $0200, X
DEX
CLC
ADD_LOOP:
ADC $0200, X
DEX
BPL ADD_LOOP
```

Indirect indexed addressing

```
LDA #$00
STA $10
LDA #$02
STA $11

LDY #$03
LDA ($10), Y
DEY
CLC
ADD_LOOP:
ADC ($10), Y
DEY
BPL ADD_LOOP
```

Indirect indexed addressing

- Must use Y
- Values are stored in little endian
- X is not available

Additional modes

- Zero page addresing
  - Uses only $00-$FF
  - Faster
  - Does not use high byte
- Indexed indirect addressing
  - Uses X
  - LDA ($10, X)

Why different modes

- Performance
- Addressing different data types
- Everything is in memory
- Typing is conceptual

Memory load and store

- LDA
  - Supports all modes
- LDX
  - Immediate, absolute, absolute indexed
- LDY
  - Immediate, absolute, absolute indexed
- Affect N and Z
- STA, STX, STY as counterparts

Register to register

- TAX
- TAY
- TXA
- TYA
- TXS
- TSX

Stack instructions

- TXS, TSX
- PHA
  - A to stack
- PHP
  - Flags to stack as a byte
- PLA
  - Pops stack to A
- PLP
  - Pops stack to P

Arithmetics

- ADC
- SBC

- INC
- DEC
  - Work on memory locations

Arithmetics

- INX
- DEX
- INY
- DEY

- Comparing:
  - CMP – operand to A
  - CPX – operand to X
  - CMY – operand to Y

Logic

- AND
- EOR
- ORA
- All work on operand and A register

- ASL
  - Shift operand left, uses C flag
- LSR
  - Shift operand right, uses C flag

Logic

- ROR, ROL
- BIT – bitwise AND to A

Branching

- JMP – Unconditional – push to PC
- BCC, BCS – jump if C clear or set
- BNE, BNQ – jump if Z clear or set
- BPL, BMI – jump if zero or not zero
- BVC, BVS – jump if V clear or set

- Conditional branching is relative, 8bit signed

Subroutines

- JSR
- RTS

Tasks

- How much space is there between $200 and $5FF in bytes?
- Fill 512 bytes of memory from location $200 with value $1
- Fill half of the memory with one and other half with another colour
- 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.

Tasks 2

- 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.

Thank you for your attention!