6502 Interrupt and Bus

Philipp Koehn

~~5 March 2018~~
23 Sept 2019

HW3 - due Friday 9/27

Midterm: in class Friday
10/4 (private msg
to instructors on Piazza
if conflict)

What Makes the Cursor Blink?

- 6502 CPU processes sequence of instructions

- But: general maintenance needed, e.g.,

  – cursor blinking

  – storage I/O

  – process key strokes

- Regular scheduled program must be interrupted

Multi-Process Operating Systems

- Modern operating systems manage multiple processes

- Scheduler in kernel switches between them

- Process state for each must be preserved

- But 6502 operation systems used simpler approach

Interrupt

- Idea

  - interrupt regularly scheduled code every once in a while

  - take care of maintenance tasks

Interrupt

- Idea

  - interrupt regularly scheduled code every once in a while

  - take care of maintenance tasks

- 6502 implementation

  - FFFE/FFFF contain address for interrupt routine
  - *on C64* triggered by a hardware clocks

  - RTI: instruction to return from interrupt ← *restores status flags*

  - SEI: disable interrupts
  - CLI: enable interrupts } *synchronize non-interrupt code w/ interrupt handlers*

Commodore C64 Interrupt Code

- Interrupt vector FFFE/FFFE points to FF48

- Save registers

Address	Bytes	Command
FF48	48	PHA
FF49	8A	TXA
FF4A	48	PHA
FF4B	98	TYA
FF4C	48	PHA

Online reference: http://unusedino.de/ec64/technical/aay/c64/romff48.htm

Software and Hardware Interrupts

- Hardware interrupt: triggered by clock

- Software Interrupt: caused by BRK instruction (sets break flag)

- Interrupt call pushes status register to stack

Software and Hardware Interrupts

- Hardware interrupt: triggered by clock

- Software Interrupt: caused by BRK instruction (sets break flag)

- Interrupt call pushes status register to stack

- Detect what kind of interrupt

Address	Bytes	Command
FF4D	BA	TSX ; get stack pointer
FF4E	BD 04 01	LDA $0104,X ; load stored status register
FF51	29 10	AND #$10 ; is the break flag set?
FF53	F0 03	BEQ $FF58
FF55	6C 16 03	JMP ($0316) ; software (BRK) interrupt
FF58	6C 14 03	JMP ($0314) ; hardware interrupt

Redirect Interrupts

- Jump to hardware interrupt routine

  Address   Bytes   Command
    FF58   6C 14 03   JMP ($0314) ; hardware interrupt

- This is a pointer

$\rightarrow$ can be modified to your own routine

Making the Cursor Blink

- Pointer at 0314/0315 points to (EA31) *default H/W int handler*

Making the Cursor Blink

- Pointer at 0314/0315 points to EA31

- Subroutine call for real time clock increment

Address	Bytes	Command
EA31	20 EA FF	JSR $FFEA ; increment Real-Time Clock

Making the Cursor Blink

- Pointer at 0314/0315 points to EA31

- Subroutine call for real time clock increment

Address	Bytes	Command
EA31	20 EA FF	JSR $FFEA ; increment Real-Time Clock

- Cursor blinking code

Address	Bytes	Command
EA34	A5 CC	LDA $CC ; is the cursor in blink mode?
EA36	D0 29	BNE $EA61 ; no → skip all this
EA38	C6 CD	DEC $CD ; count down cursor blink timer
EA3A	D0 25	BNE $EA61 ; not at 0 → done
EA3C	A9 14	LDA #$14
EA3E	85 CD	STA $CD ; reset timer

Making the Cursor Blink

```
Address    Bytes        Command
   EA40    A4 D3        LDY $D3        ; cursor column
   EA42    46 CF        LSR $CF        ; currently solid → set carry
   EA44    AE 87 02     LDX $0287      ; load color under cursor
   EA47    B1 D1        LDA ($D1),Y    ; retrieve character from screen memory
   EA49    B0 11        BCS $EA5C      ; earlier check:  branch if solid
   EA4B    E6 CF        INC $CF        ; set cursor status "solid"
   EA4D    85 CE        STA $CE        ; store character value under cursor
   EA4F    20 24 EA     JSR $EA24      ; synchronize color pointer
   EA52    B1 F3        LDA ($F3),Y    ; load color from screen color memory
   EA54    8D 87 02     STA $0287      ; set color under cursor
   EA57    AE 86 02     LDX $0286      ; get current color
   EA5A    A5 CE        LDA $CE        ; get character code under cursor
   EA5C    49 80        EOR #$80       ; invert character code
   EA5E    20 1C EA     JSR $EA1C      ; print character to screen
```

Return from Interrupt

- Restore registers

Address	Bytes	Command
EA81	68	PLA
EA82	A8	TAY
EA83	68	PLA
EA84	AA	TAX
EA85	68	PLA

- Return from interrupt

EA86   40   RTI   } restores status flags

Example: Redirect Interrupt

- Border color is set in D020

- Change border color at each interrupt

Address	Bytes	Command
4000	78	SEI ; no interrupt while we change pointer
4001	A9 0D	LDA #0D
4003	8D 14 00	STA 0314
4006	A9 40	LDA #40
4008	8D 15 03	STA 0315 ; redirect interrupt to 400D
400B	58	CLI
400C	00	RTS *BRK*
400D	EE 20 D0	INC D020
4010	4C 31 EA	JMP EA31 ; jump to regular interrupt reoutine

(handwritten annotations: "address 400D", "BRK", arrows)

bus
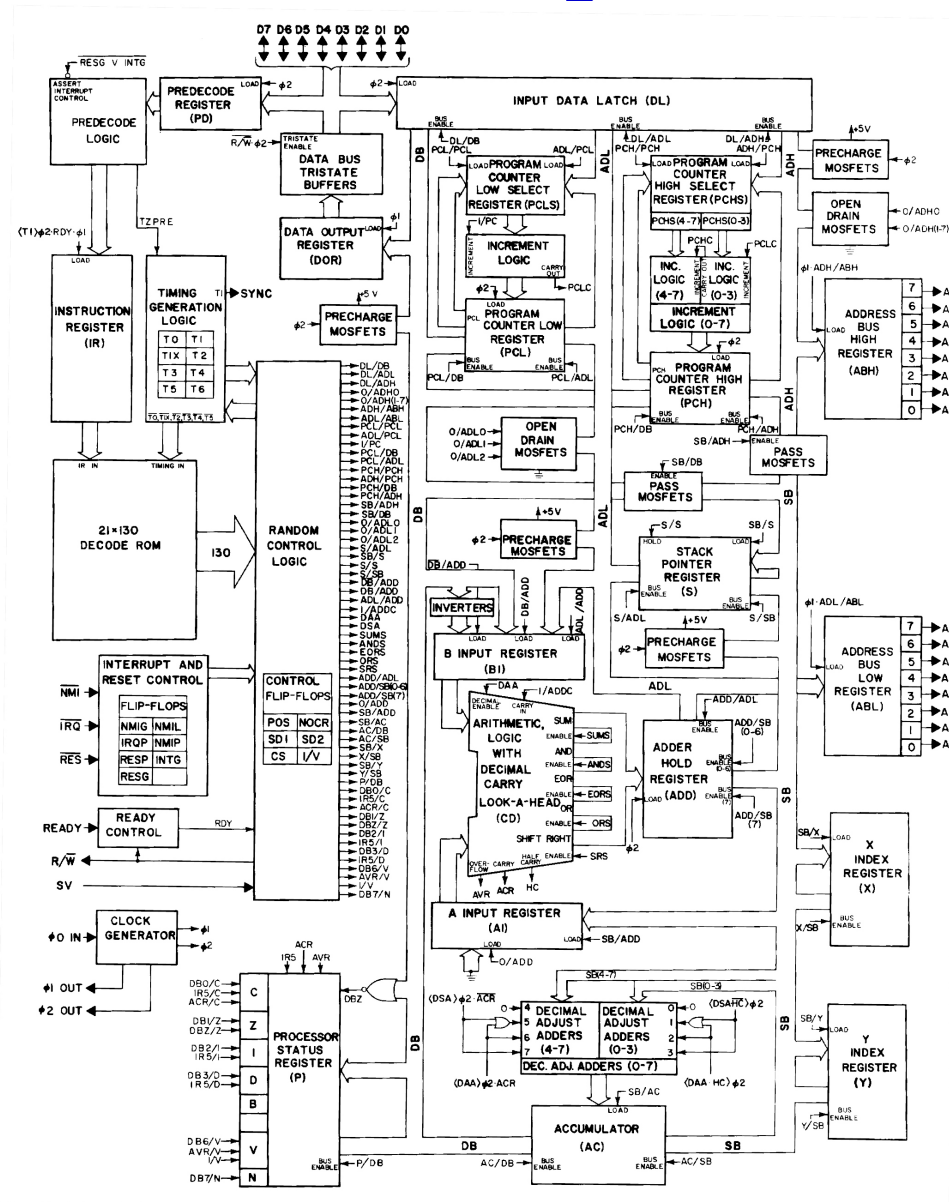
Many Components, Many Data Paths

- Problem: Increasing number of components

  – multiple registers
  – program counter
  – stack pointer
  – arithmetic logic unit
  – memory

- More components, more data paths

- Some components are outside the CPU

  – main memory
  – video processing
  – keyboard
  – tape drive and other storage

} HW devices

- Bus: shared data paths

- Example: data bus connects

  - accumulator

  - program counter

  - status register

  - arithmetic logic unit (ALU)

  - also (indirectly) connected to pins of chip

- Microprogram instructions

  - AC/DB: place accumulator value on bus
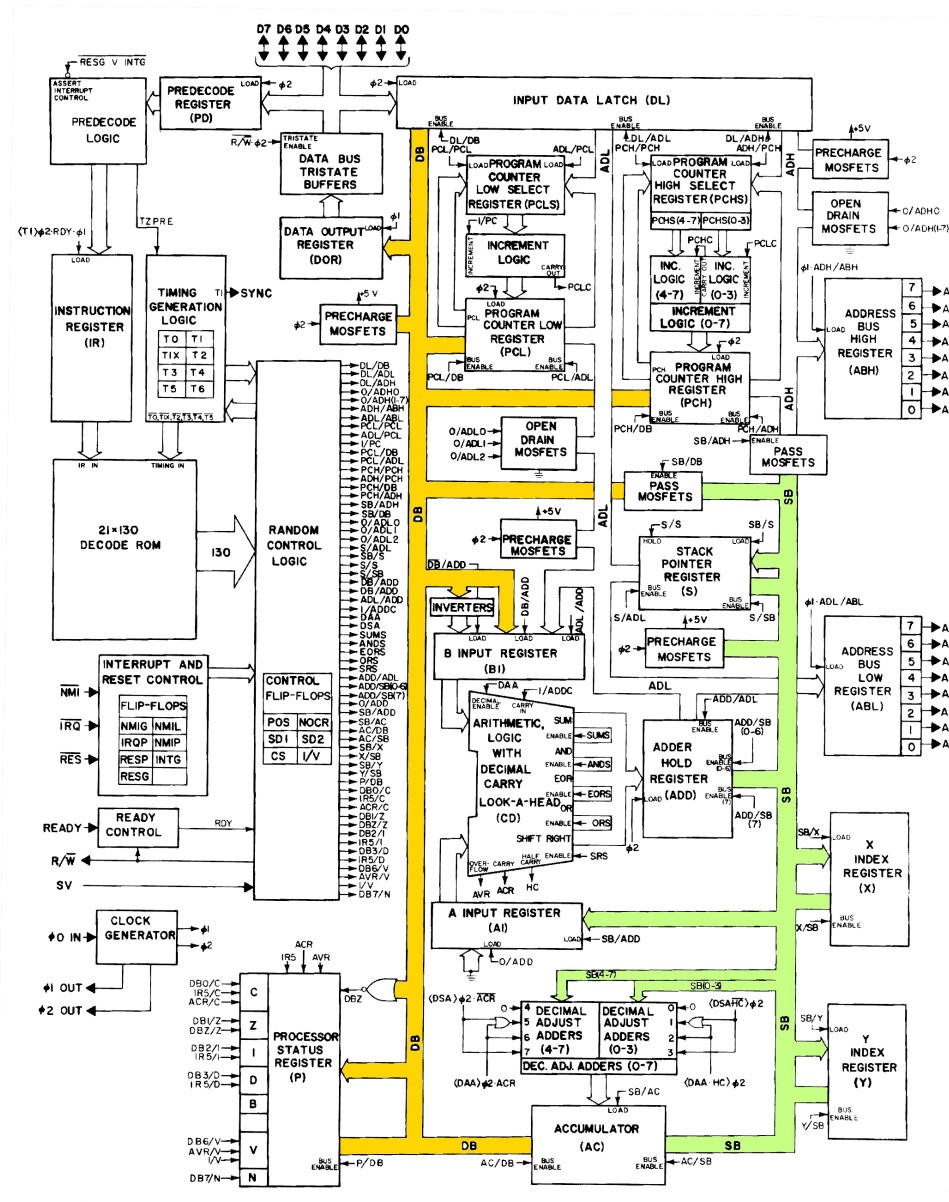
  - DB/ADD: read add input to ALU from bus

Data Bus (DB)
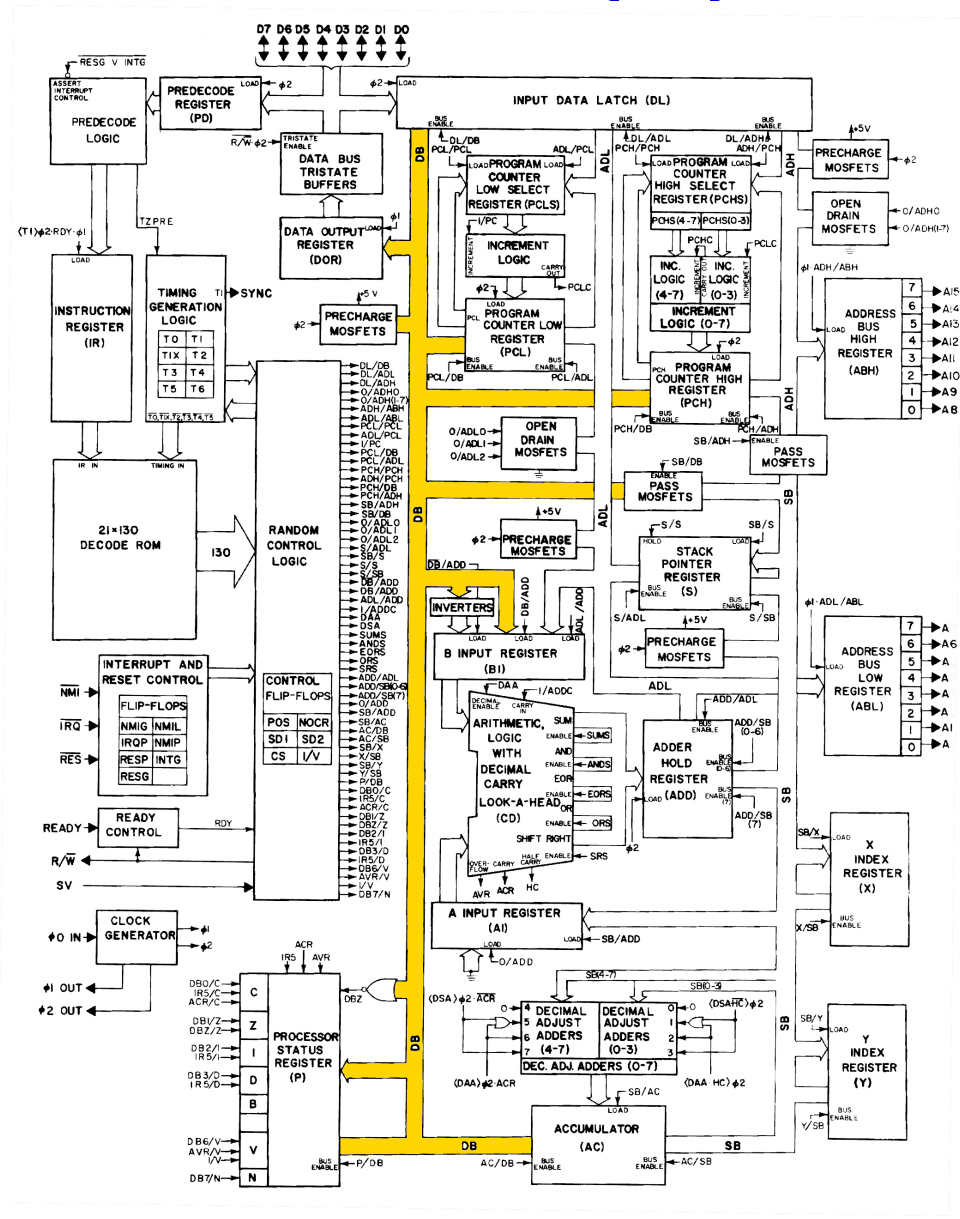
Internal Buses

— externally visible!

- Data bus (DB)

  – accumulator

  – program counter

  – status register

  – arithmetic logic unit (ALU)

  – also (indirectly) connected to pins of chip

- Special bus (SB)

  – registers (A, X, Y)

  – arithmetic logic unit (ALU)

  – stack pointer

  – only internal

Address Bus (ABL/ABH)

externally visible!

- Addresses are 16 bits

- Memory is based on 8 bit
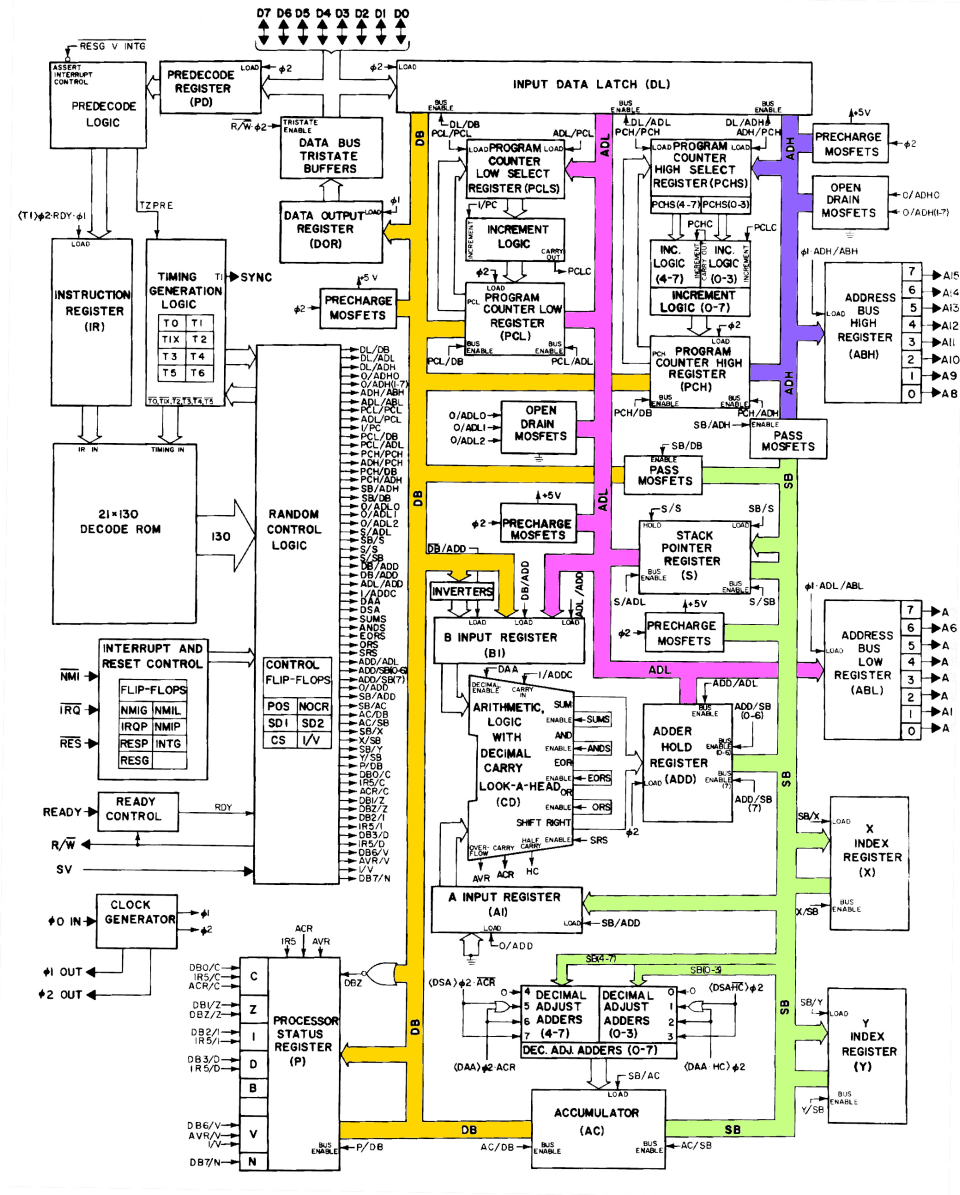
⇒ 2 buses for memory addresses

  – ABL: address bus low
  – ABH: address bus high

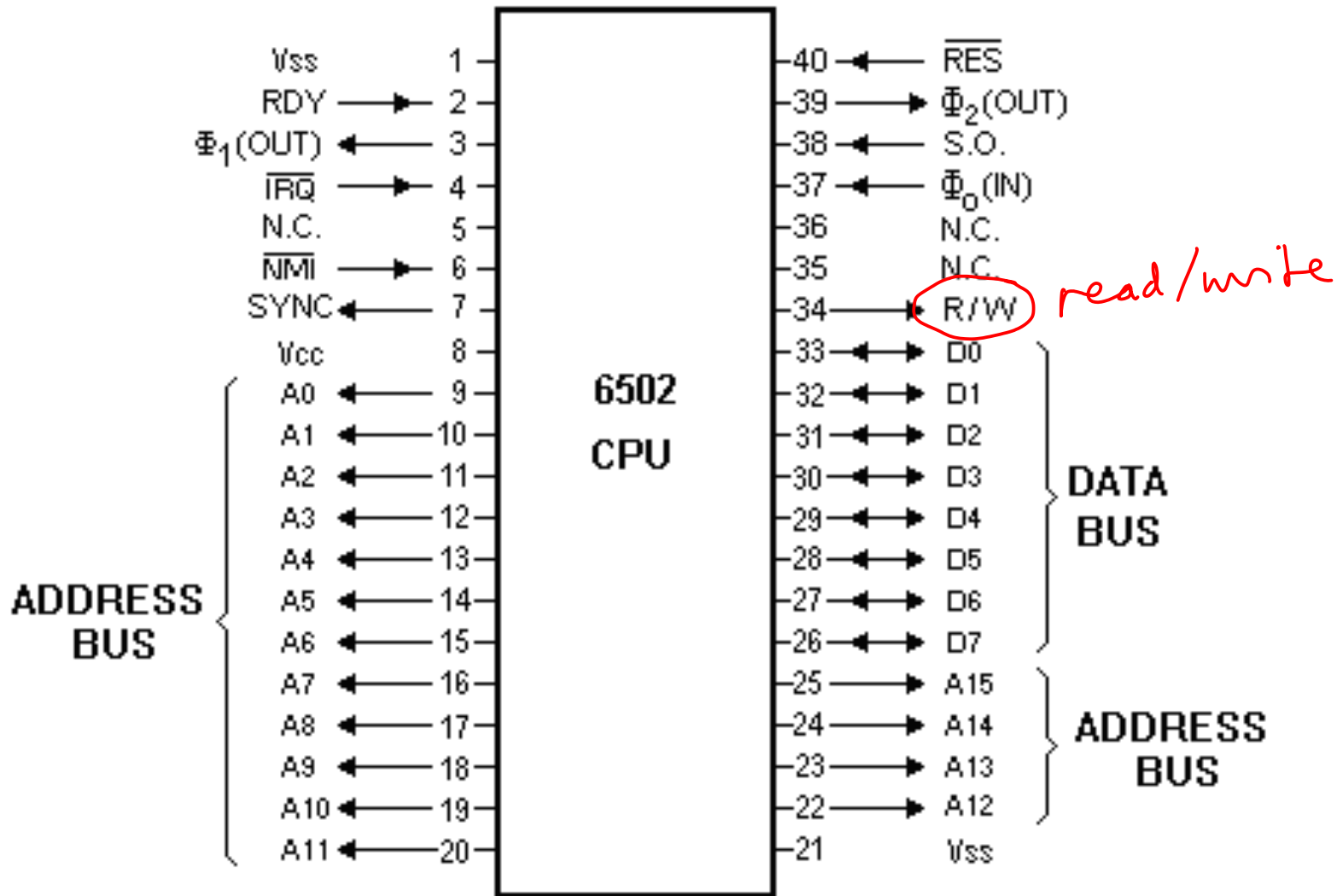- Pin connection to memory (outside CPU)

Internal vs External Bus

- Internal bus

  – operates within the same motherboard

  – 6502: data and special bus

  – system bus connects CPU and memory

- External bus

  – connects to external devices

Universal Serial Bus (USB)

- Popular bus today: universal serial bus (USB)

- Breaks up data into series of bytes

- Protocol defines what each message means

- Also contains power line
  $\rightarrow$ cell phone charging

Bus Speed

- Different buses may operate at different speeds

  - within same chip: very fast

  - on same motherboard: fast

  - external bus: slowest

- Some speed numbers today

  - CPU speed: 3 GHz

  - System bus (CPU to memory): ~~100-200 MHz~~ 3 GHz

  - USB 3.0 (2013) maximum speed: up to 2.4 GHz