BOBS: an Intel 8085 processor simulator

Antônio Drumond Cota de Sousa (antonio.drumondcs@gmail.com),
Mateus Henrique Medeiros Diniz, Davi Ferreira Puddo,
Henrique Cota de Freitas

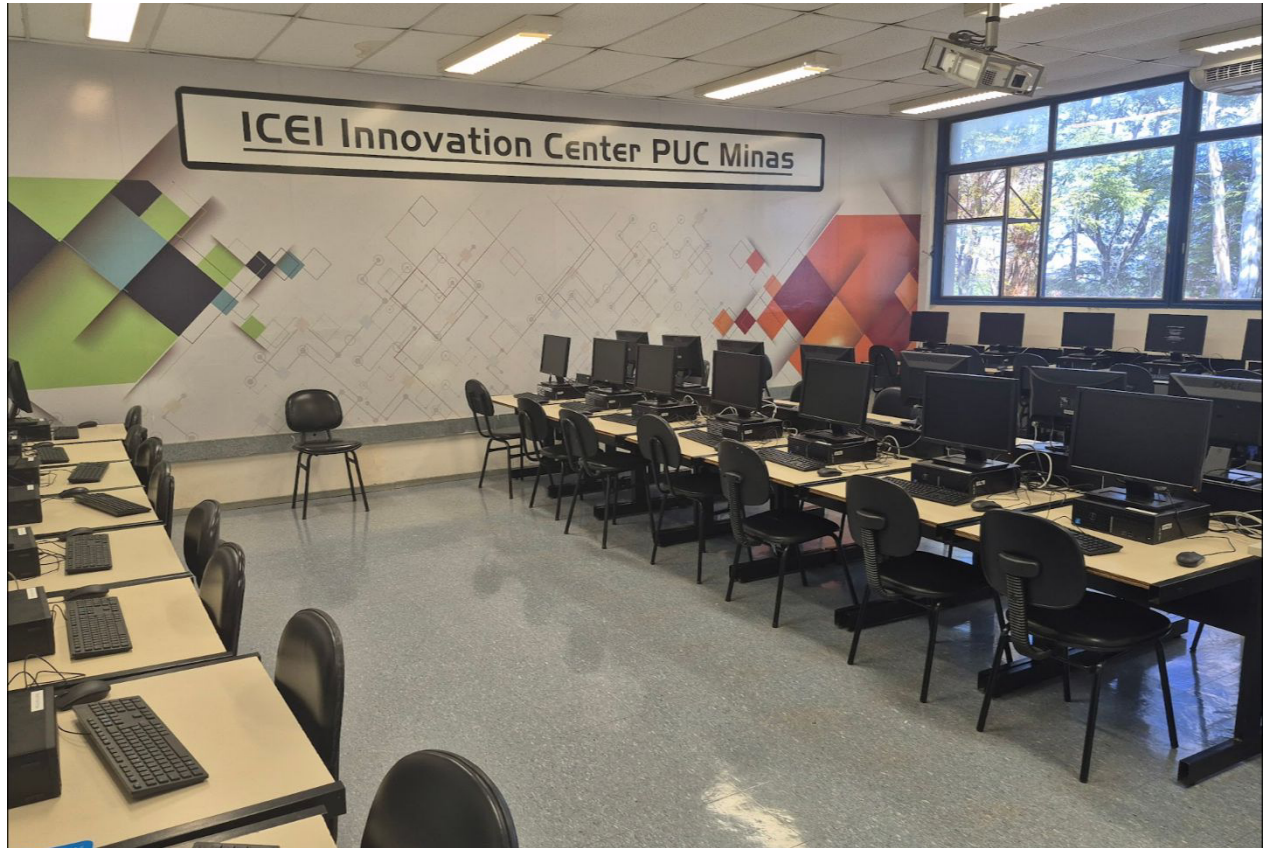
PUC Minas

ERAD SUDESTE 2025

Outline

- Introduction
- Related Work
- The 8085 Processor
- BOBS Simulator
- Conclusion

Introduction: Processor simulator



Why a simulator?

- Accessibility
- Automation
- Ease of use
- No cost

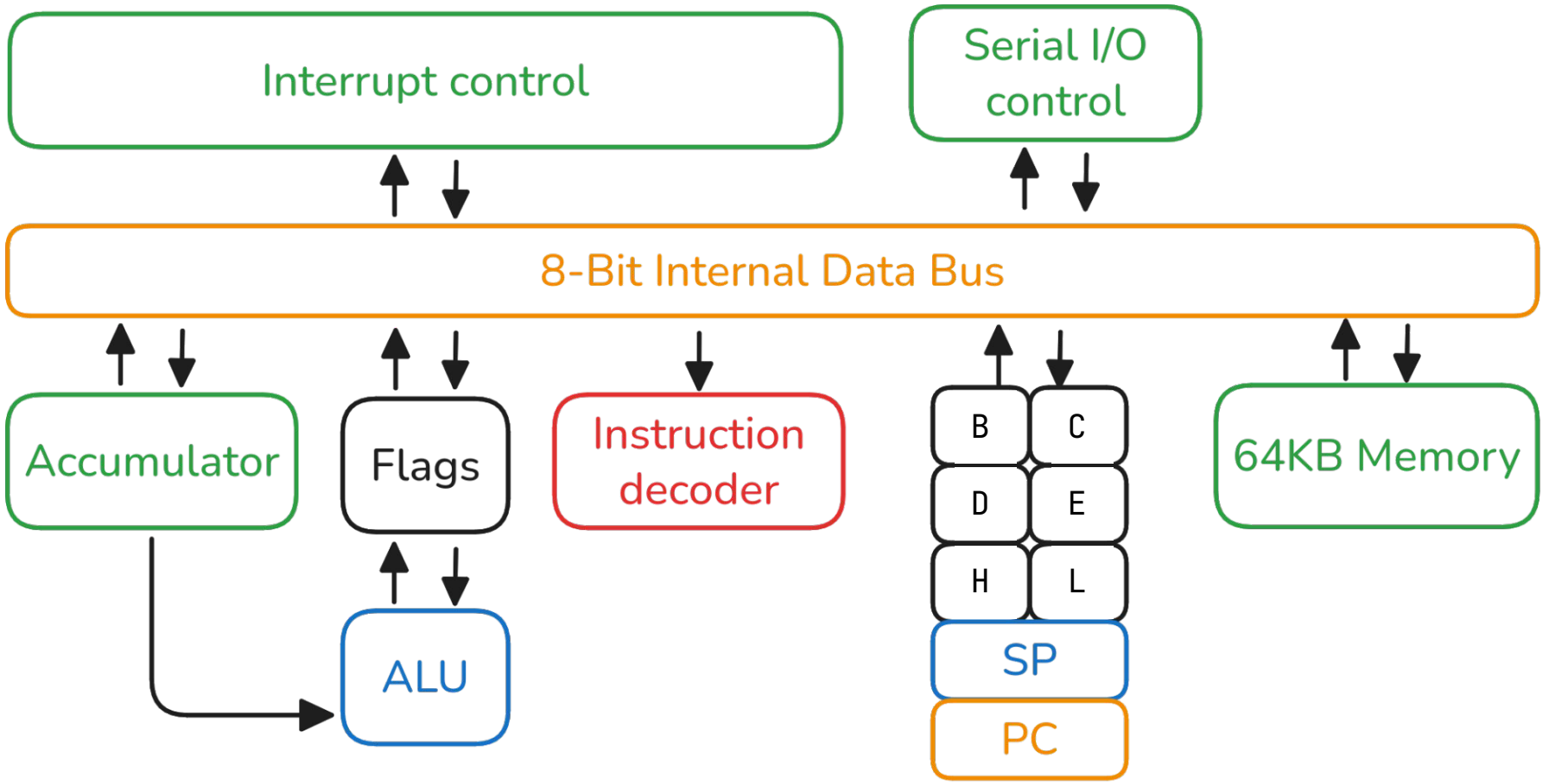
PUC Minas

Related Work

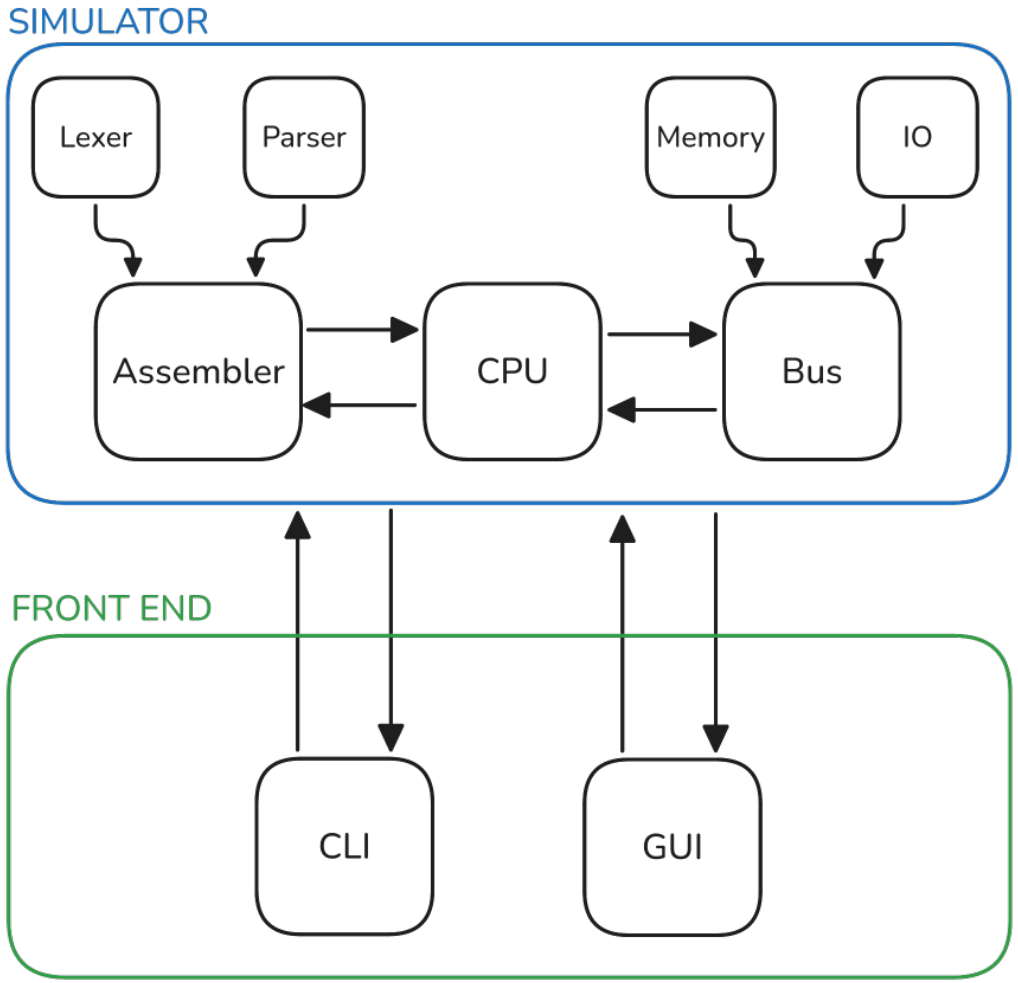
Simulator	Platform	CLI Interface	GUI Interface	Simulated architecture
MARS Vollmar and Sanderson (2006)	Windows, Linux and MacOS	☑️	✅	MIPS
Sparrow Liang et al. (2023)	Windows	✅	✅	Generic architecture from textbook “Computer Composition Principle”
Tomasulo's Algorithm Simulator Nolasco et al. (2023)	Source code Only	✅	❌	Generic RISC-V like architecture
BOBS	Windows, Linux and MacOS	✅	✅	Intel 8085

PUC Minas

The 8085 Processor



BOBS Simulator



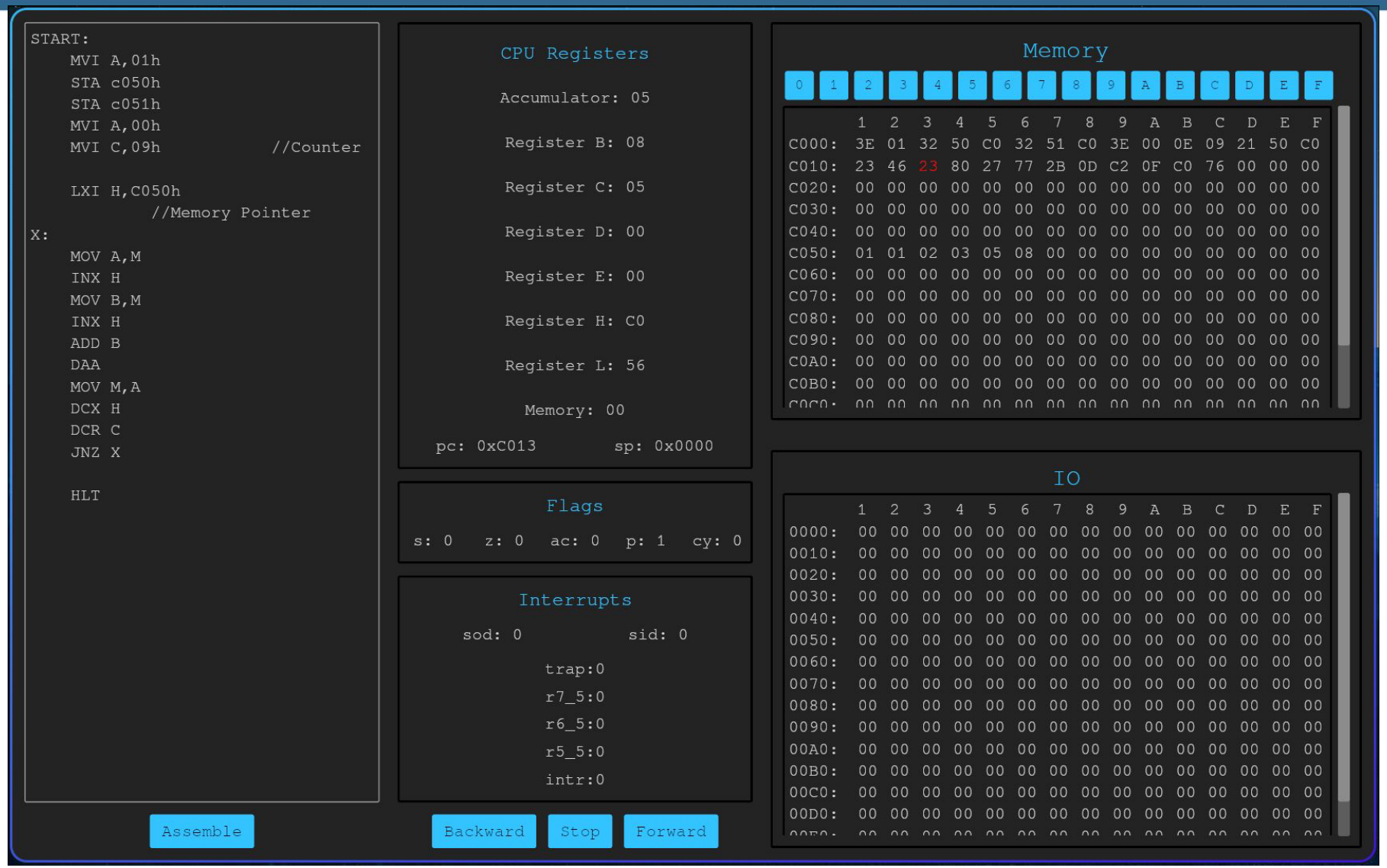
BOBS Simulator

```
> $ h
BOBS_8085   v1.2.2
Developed by Antônio Drumond, Mateus Diniz and Davi Puddo

Simulator commands:
h | help                    ⟶ Prints this
clear | cls                 ⟶ Clear terminal screen
exit | quit | q             ⟶ Exit simulator
run [FILENAME]              ⟶ Assemble and run (Without step) program in file
run step [FILENAME]         ⟶ Assemble and run (Step by step) program in file
run bin [FILENAME]          ⟶ Run program from binary memory file
run bin step [FILENAME]     ⟶ Run program (Step by step) from binary memory file
                           (Program in binary memory file should be between positions
                            C000 and CFFF in memory)
assemble [INPUT] [OUTPUT] ⟶ Assemble program in plain text file([INPUT]) and creates
                                memory file ([OUTPUT])

> $
```

```
step: 35

📥A  ⇒ 05 - 00000101    |    🚩S  ⇒ false
📥B  ⇒ 03 - 00000011    |    🚩Z  ⇒ false
📥C  ⇒ 06 - 00000110    |    🚩AC ⇒ false
📥D  ⇒ 00 - 00000000    |    🚩P  ⇒ true
📥E  ⇒ 00 - 00000000    |    🚩CY ⇒ false
📥H  ⇒ C0 - 11000000
📥L  ⇒ 53 - 01010011
📥SP ⇒ 0000 - 0000000000000000
📥PC ⇒ C018 - 1100000000011000


Memory saved to "./memory.txt"
IO ports saved to "./io.txt"
Options:

[F]/[Forward]/[>]  ⇒ Go forward 1 step

[S]/[Stop]/[Exit]/[|]  ⇒ Exit step by step execution

[B]/[Backward]/[<]  ⇒ Go back 1 step

[P]/[Print]/[Print + range]  ⇒ Print the memory

> $
```

PUC Minas

BOBS Simulator

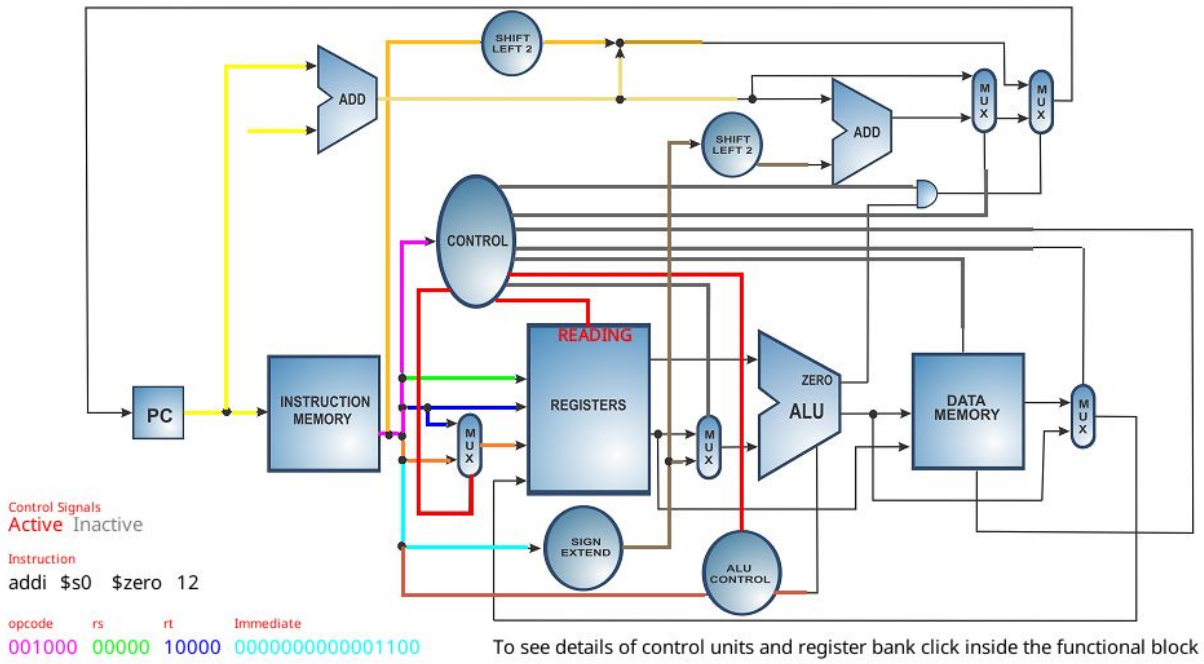


Conclusion

Our simulator

- Teaching features
- GUI and CLI interfaces
- Entire Intel 8085A instruction set
- Built in rust

Future Work



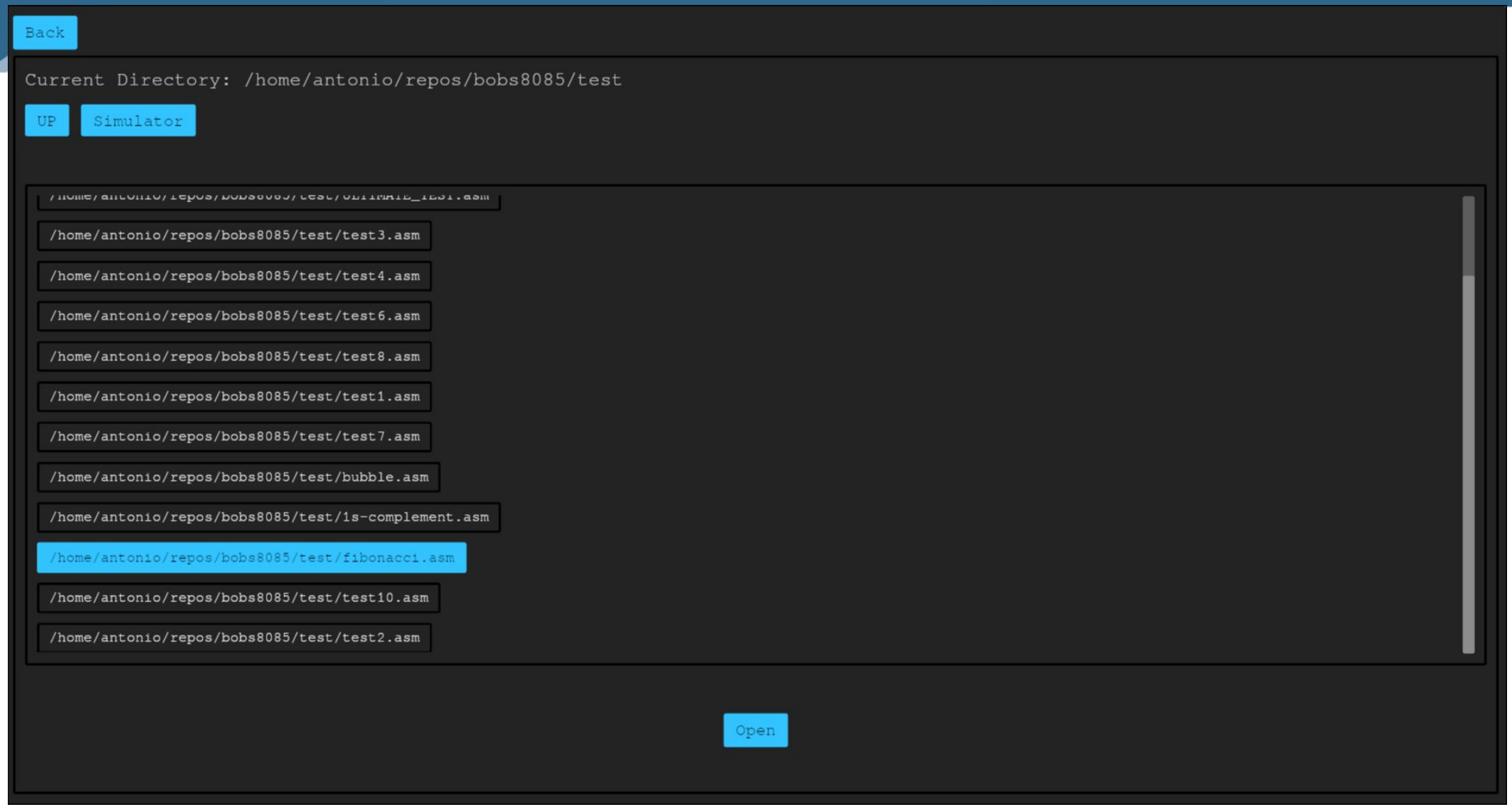
Pipeline view from MARS Simulator Vollmar and Sanderson (2006)

Possible features

- Real time code linting and completion
- Pipeline view
- Peripherals support

Thanks

Appendix 1


Back
Current Directory: /home/antonio/repos/bobs8085/test
UP
Simulator
/home/antonio/repos/bobs8085/test/test3.asm
/home/antonio/repos/bobs8085/test/test4.asm
/home/antonio/repos/bobs8085/test/test6.asm
/home/antonio/repos/bobs8085/test/test8.asm
/home/antonio/repos/bobs8085/test/test1.asm
/home/antonio/repos/bobs8085/test/test7.asm
/home/antonio/repos/bobs8085/test/bubble.asm
/home/antonio/repos/bobs8085/test/1s-complement.asm
/home/antonio/repos/bobs8085/test/fibonacci.asm
/home/antonio/repos/bobs8085/test/test10.asm
/home/antonio/repos/bobs8085/test/test2.asm
Open

Appendix 2

Back | Arithmetic | Branching | Control | Data Transfer | Logical

ADD SRC
    Name: Add Register or Memory to Accumulator
    Description: The 8-bit contents of the SRC register (A, B, C, D, E, H, L, M) are added to the 8-bit contents of the accumulator, and the result is stored in the accumulator. If M is specified, the operand is fetched from the memory location specified by the HL register pair.
    Flags: All flags (S, Z, AC, P, CY) are affected.

ADC SRC
    Name: Add Register or Memory to Accumulator with Carry
    Description: The 8-bit contents of the SRC register (A, B, C, D, E, H, L, M) and the Carry Flag (CY) are added to the 8-bit contents of the accumulator. The result is stored in the accumulator. If M is specified, the operand is fetched from the memory location specified by the HL register pair.
    Flags: All flags (S, Z, AC, P, CY) are affected.

ADI DATA
    Name: Add Immediate to Accumulator
    Description: The 8-bit immediate DATA is added to the 8-bit contents of the accumulator, and the result is stored in the accumulator.
    Flags: All flags (S, Z, AC, P, CY) are affected.

ACI DATA
    Name: Add Immediate to Accumulator with Carry
    Description: The 8-bit immediate DATA and the Carry Flag (CY) are added to the 8-bit contents of the accumulator. The result is stored in the accumulator.
    Flags: All flags (S, Z, AC, P, CY) are affected.

DAD REG_PAIR
    Name: Add Register Pair to H and L Registers
    Description: The 16-bit contents of the specified register pair (B, D, H, or SP) are added to the 16-bit contents of the HL register pair. The result is stored in the HL register pair.

PUC Minas

Appendix 3

```
Accumulator: 03
 Register B: 03
 Register C: 06
 Register D: 00
 Register E: 00
 Register H: C0
 Register L: 54
   Memory: 05
pc: 0xC011          sp: 0x0000
```

Memory

0 1 2 3 4 5 6 7 8 9 A B C D E F

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
C000: 3E 01 32 50 C0 32 51 C0 3E 00 0E 09 21 50 C0 7E
C010: 23 46 23 80 27 77 2B 0D C2 0F C0 76 00 00 00 00
C020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C050: 01 01 02 03 05 00 00 00 00 00 00 00 00 00 00 00
C060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

PUC Minas