

Operating Systems

CMPT 424 • Fall 2024

– iProject Two - 50 points

Goals	To build on the functionality from iProject One (all of which is required) by adding the ability to load and execute one user program in 256 bytes of memory as specified by the 6502a op codes documented on our class web site.
Functional Requirements	<ul style="list-style-type: none"><input type="checkbox"/> Modify the load command to copy the 6502a machine language op codes into main memory. [5 points]<ul style="list-style-type: none">• Put the code at location \$0000 in memory• assign a Process ID (PID)• create a Process Control Block (PCB)• return the PID to the console and display it.<input type="checkbox"/> Add a shell command, run <pid>, to run a program already in memory. [3 points] Note: the user should be able to execute many load/run cycles in sequence.<input type="checkbox"/> Execute the running program (including displaying memory, CPU, and PCB status as well as any output). Be sure to synchronize the CPU execution cycles with clock ticks. [15 points]<input type="checkbox"/> As the programs executes, update and display the memory, PCB, and CPU status (program counter, instruction reg, accumulator, X reg, Y reg, Z flag) in real time. [5 points]<input type="checkbox"/> Implement line-wrap in the CLI. (This is not longer optional.) [2 points]<input type="checkbox"/> [challenge] Provide the ability to single-step execution (via GUI buttons). [+4 points]<input type="checkbox"/> [challenge] Allow the user to break the current program with Ctrl-C. [+2 points]
Implementation Requirements	<ul style="list-style-type: none"><input type="checkbox"/> Develop a PCB prototype and implement it in the client OS. [4 points]<input type="checkbox"/> Develop a memory manager and implement it in the client OS. [4 points]<input type="checkbox"/> Develop a core memory prototype and implement it in the host OS. [4 points]<input type="checkbox"/> Develop a memory accessor prototype and implement it in the host OS. [4 points]<input type="checkbox"/> Develop a CPU prototype and implement it in the host OS. [4 points]<input type="checkbox"/> Your code <i>must separate structure from presentation</i>, be professionally formatted, use and demonstrate best practices, and and be free of compiler errors. If there are compiler errors it's -10 * project number points off. [-∞ if not]<input type="checkbox"/> Continue to preserve GLaDOS and link the correct testing code.<input type="checkbox"/> Continue to cite everything that's not entirely your own original work.
General Hints	<p>Consider refactoring the host OS code into at a few parts: events, interrupts, and services.</p> <p>Implement the host core memory and CPU as a separate objects and separate source code files within the host directory.</p> <p>Run all accesses to memory through your host memory accessor so it can do address translations on the next project.</p> <p>Read chapter 4 up to “Paging” in our text. It presents a nice look at main memory that you might find helpful.</p> <p>Remember the utility of comments and how much their presence and quality affect my opinion of your work. Seriously. Bad code style will cost you a ton of points.</p>

Operating Systems

CMPT 424 • Fall 2024

Specific
Hints

- Continue to keep .js and .ts files in separate directories.
- The shutdown functionality must operate properly even when a program is running.
- The memory and CPU displays must update in real time. The PCB needs to update only on scheduling events, which in this project are only when execution begins and ends.
- Remember that you must continue to process interrupts while the CPU is executing.
- Always separate logic and structure from presentation. **Do not** put GUI logic in your CPU or memory routines. Keep the GUI stuff in control.ts or something similar.
- As I warned you in class, it's easy to mix up base 10 and base 16. Double check all of that.
- You are almost guaranteed to make an off-by-one error implementing or executing the BNE operation. Embrace it. Then fix it.
- Never use magic numbers. Use constants for everything other than 0, 1, and ∞ .

Very Specific
Hints

Follow the CPU example when implementing memory.

In the main directory:

```
Declare globally...
// Hardware (host)
var _CPU: TSOS.CPU;
var _Memory: TSOS.Memory;
var _MemoryAccessor: TSOS.MemoryAccessor;
// Software (OS)
var _MemoryManager: any = null;
```

In the host/ directory, put:

```
cpu.ts
memory.ts
memoryAccessor.ts
```

In the hardware control code :

```
_CPU = new Cpu();
_CPU.init();
_Memory = new Memory();
_Memory.init();
_MemoryAccessor = new MemoryAccessor();
```

In the OS/ directory, put :

```
memoryManager.ts
```

In the kernel bootstrap code:

```
_MemoryManager = new MemoryManager();
```

Submitting
Your Work

Update GitHub with your current code and let me know that it's ready for me to grade (and which branch to grade).

Operating Systems

CMPT 424 • Fall 2024

```
>load
Process ID: 0
>run 0
>12
```

Log

Sun, Aug 28th 2016, 1:33:05 pm	900
OS Idle	
Sun, Aug 28th 2016, 1:32:43 pm	474
OS CPU cycle	

Processes

Round Robin

PID	PC	IR	ACC	X	Y	Z	Priority	State	Location
0	34	A9	79	3	2	1	32	Running	Memory

Memory

0x020	A9	4F	8D	43	00	A9	4E	8D
0x028	44	00	A9	45	8D	45	00	A9
0x030	00	8D	46	00	A2	02	A0	42
0x038	FF	00	00	00	00	00	00	00

CPU

LDA #\$4F

PC	IR	ACC	X	Y	Z
022	A9	79	3	2	1

```
A9 03 8D 41 00 A9
01 8D 40 00 AC 40
00 A2 01 FF EE 40
00 AE 40 00 EC 41
00 D0 EF A9 44 8D
42 00 A9 4F 8D 43
00 A9 4E 8D 44 00
A9 45 8D 45 00 A9
00 8D 46 00 A2 02
A0 42 FF 00
```

OS iProject 2

© 2008-2112 Alan G. Labouseur, All Rights Reserved

Page 3 of 3