Microcode Compiler Settings

The compiler needs to know the size of the ROM chips you're using. Enter the number of bits for the address and the number of bits for the data.

For example, an AT28C256 is a 32Kx8 EEPROM. It has 15 bits for the address and 8 bits for the data.

ROM Chip Address Bits:	13
ROM Chip Data Width:	8

Specify the output file name. Include a number sign (#) which will be used to index multiple files. Files will be written to your Google Drive.

Output File Name Pattern:	microcode#.rom
File Format:	ROM Binary

IMPORTANT: For the script to run, you need to enable the Drive API.

To do this, select the Extensions->App Script menu. A new tab will open up with the compiler's code in it.

In the left column, click on Services. A dialog will appear. Scroll down to "Drive API". Select it, then click the Add button.

You can then close the browser tab.

Compile

	Address (MSB..LSB)							Data (MSB..LSB, ROM 1..ROM n)																		
Field	Istr Reg Bit 0	IRQ	Running	Unused	Condition True	Op Code	Cycle	MC ALU Op Code (12 means use IR)	Reg Left Addr (0 means use IR, 3 means R15, 2 means R14, 1 means R13)	Reg Left R/~W	Reg Left ~OE	Reg Right ~OE and ~CE	Reset Cycle Counter	Set State (1 is Running , 2 is Int Hdlr, 3 is Exit Int Hdlr)	Data Bus Latch ~Enable	Data Bus Latch ~OE	Addr Bus Latch ~Enable	Addr Bus Latch ~OE	Instr. Reg Latch ~Enable	CC Latch ~Enable	IR Immediate 8 Bits ~OE	IR Immediate 4 Bits ~OE	Mem R/~W	Mem ~OE	Unused	Comment
Bit Width	1	1	1	2	1	4	3	4	2	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	2	
Default Value	x	0	1	x	x	x	x	12	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	0	
Startup		x	0			x	0	6	3	1	0				0	1	0	1								Set ALU to output 0
		x	0			x	1	6	3	0	1				1	0	1	0								Latch the 0 onto the data bus. Write to R15
		x	0				2	6	3	1						0		0								Finish the write to R15
		x	0			x	3		1	0						0		0								Write the 0 to R13 (zero register) too
		x	0			x	4		1	1			1	1		0		0								Finish the write; set running state; reset cycle counter
Instruction Fetch						x	0	7	3	1	0						0	0								Read PC, set ALU to output A, route it to the address bus
						x	1											0	0				1	0		Read memory, latch the instruction into the instruction register
						x	2	0	3	1	0				0		0									Read the PC, add 1, latch it onto the address and data buses
						x	3		3	0						0		0								Write the incremented PC back to the PC
Load Immediate 8 bit						0	4	15		0			1		0	0					0					Route Immediate value to ALU B, set ALU to output B, write to register, and reset cycle counter
LD Ra,#imm																										
Software Interrupt	0					1	4	13	2	1	0			2	0	1	0	0								Put SP minus one on the address bus and data latch
SWI #n	0					1	5		2	0	1				1	0										Write SP-1 to SP
n must be even	0					1	6	7	3	1	0				0	0							0			Put PC on the data bus and start a memory write
	0					1	7	7	3	0			1		0	0					0					Put immediate data on the data bus and write it to PC
Return from Interrupt	1					1	4	7	2	1	0						0	0					1	0		Put SP on the address bus. Start a memory read.
RTI	1					1	5		3	0							1	0					1	0		Hold the address on the bus. Start a write into PC
	1					1	6	0	2	1	0				0	0										Put SP+1 on the data bus
	1					1	7		2	0			1	3									1	0		Write SP+1 into SP
Load Indirect 4 bit offset						2	4	15		1	0				0											Put 4 immediate bits on ALU B, set ALU to output B, latch it into data latch. Ra is read but not used.
LD Ra,off[Rb]						2	5	3				0			1	0	0									Add the 4 bit immediate value, which is now on the data bus, to Rb and latch it onto the Address bus. Start a memory read.
						2	6			0			1			1	1	0					1	0		Read the memory and write it to Ra
Store Indirect 4 bit offset						3	4	3		1	0						0	0				0				Read Ra, add the 4 bit immediate value, latch onto address bus
ST off[Ra],Rb						3	5	15				0	1		0	0		0					0			Read Rb, write to memory, reset cycle counter
ALU op w/save to Ra						4	4			1	0	0			0					0						Read Ra and Rb, latch ALU output to data bus, latch condition codes
XXX Ra,Rb						4	5			0			1			0										Put latched data on bus, write to Ra, reset cycle counter
ALU op w/o save to Ra						5	4			1	0	0	1							0						Read Ra and Rb, latch condition codes, reset cycle counter
XXX Ra,Rb																										
ALU op w/immediate value 16 bit						6	4					0			0			0					1	0		PC was left on the addr bus by the instruction load. Read the immediate value from memory, read right reg, do ALU op, latch it onto the data bus
XXX Ra,Rb,#n16						6	5			0		0				0	1	0					1	0		Save the ALU result in left reg
						6	6	0	3	1	0				0											Add one to the PC
						6	7	0	3	0			1		1	0										Save the incremented PC
Call	0					7	4	13	2	1	0				0	1	0	0								Put SP minus one on the address bus and data latch
CALL Rb	0					7	5		2	0	1				1	0										Write SP-1 to SP
	0					7	6	7		1	0				0	0							0			Put Ra on the data bus and start a memory write (for a CALL, Ra is the PC)
	0					7	7	15		0		0	1		0	0										Put Rb on the data bus and write it to Ra (Ra must be PC)
Pop/Return	1					7	4	7	2	1	0						0	0					1	0		Put SP on the address bus. Start a memory read.
POP Ra	1					7	5			0							1	0					1	0		Hold the address on the bus. Start a write into Ra
	1					7	6	0	2	1	0				0	0										Put SP+1 on the data bus
	1					7	7		2	0			1										1	0		Write SP+1 into SP
Branch, condition true					1	8	4	3	3	1	0				0											PC to ALU A, immediate data to ALU B, ALU op A+B, latch result into data
Bcc offset					1	8	5		3	0			1			0										Write latched data to PC, reset cycle counter
Branch, condition false					0	8	4						1													Reset cycle counter
Bcc offset																										

Compilation complete.
ROM required: 8Kx8; 3 chips.