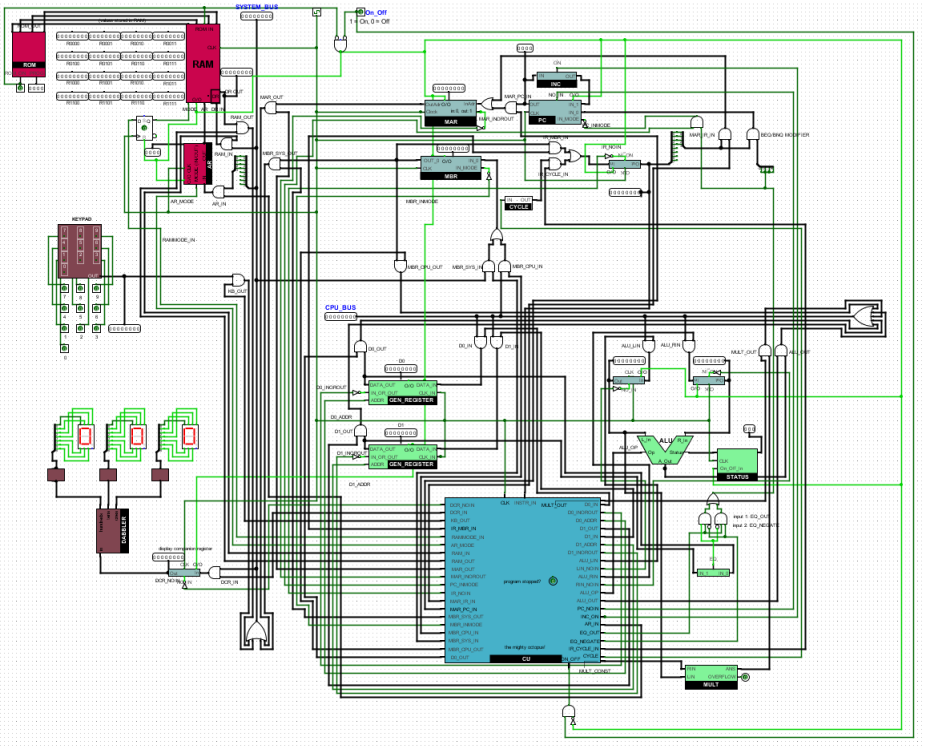
A Comprehensive Guide to the OK-Computer

Julian Lore (260741656)
Sandrine Monfourny-Daigneault (260692380)
Jacques Vincent (260744151)

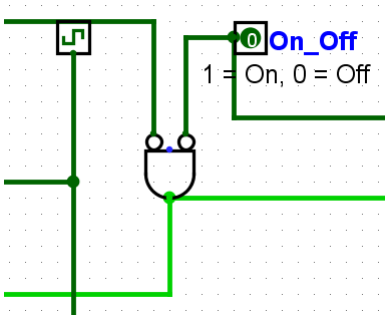
April 10th, 2017



Contents

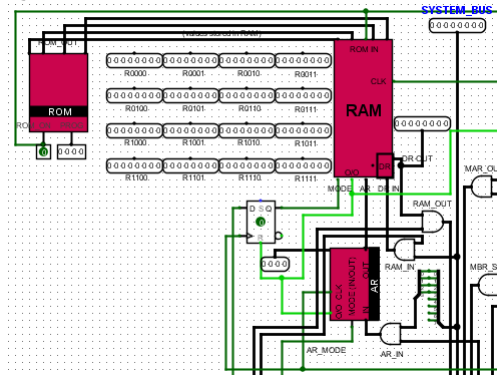
CPU Structure

Power



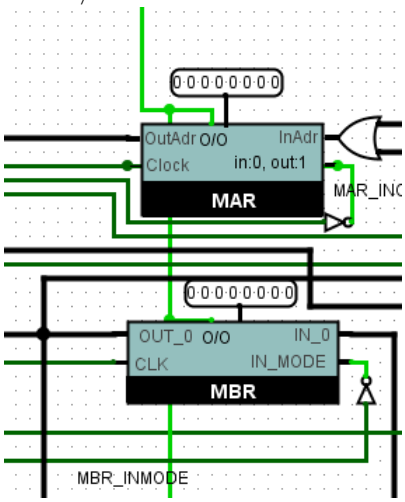
The on button allows the CU to operate and perform instructions. When it is off and the ROM is also off, all data of the CPU is reset.

RAM



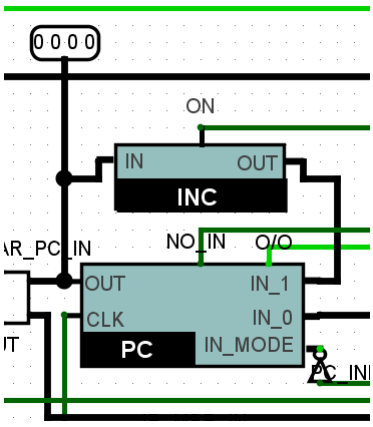
The RAM is connected to the ROM and its 3 respective registers, the mode, data and address registers. The ROM acts as an override to the usual ROM registers, OR gating the input of each byte in order to fill up all of RAM in one tick. The output pins next to the RAM indicate the values that all the bytes of RAM are storing. The mode register is controlled by the CU and the data and address register receive input from the bus, with the CU controlling the gates that let data through.

MAR/MBR



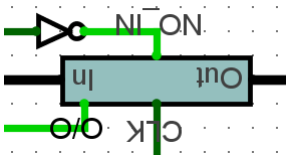
The MAR and MBR have corresponding output pins displaying their contents. They have an in/out input that controls whether they are outputting data or inputting data. They output data by default in order to not intake 0s when nothing is specifically being sent to them. The MAR receives input either from the IR (for instruction arguments) or the PC (in order to load the corresponding address' instruction). The MBR receives input from either of the data registers, or from RAM through the bus.The MAR outputs to RAM's AR through the bus, while the MBR outputs to either the RAM's DR through the bus, any of the general registers or the IR.

PC & INC



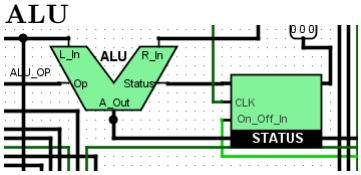
The PC's incrementer increments whenever the CU reports that it is done an instruction and thus updates the PC's contents. The PC also receives an address from the IR if BEQ or BNEQ results in true. The PC also has an in/out mode.

IR



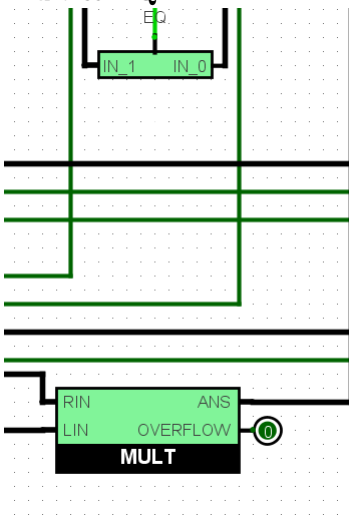
The IR receives data from the MBR whenever the CU requests a new instruction. The IR delivers its op-code and operands to the CU for parsing.

ALU



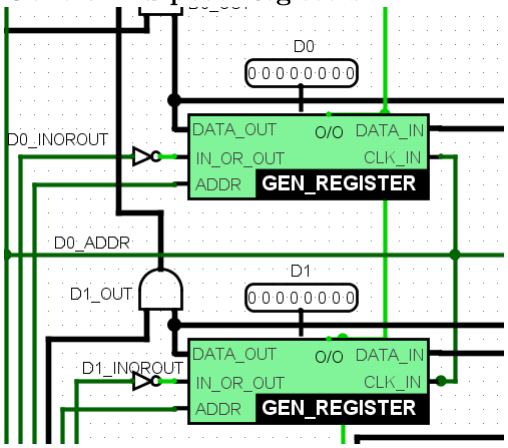
The ALU receives arguments from the general data registers and performs the operation specified by the op code given by the CU. It puts its answer in A Out which goes to the specified general purpose register and has a status register that will specify whether there were any errors with the operation.

Mult & EQ



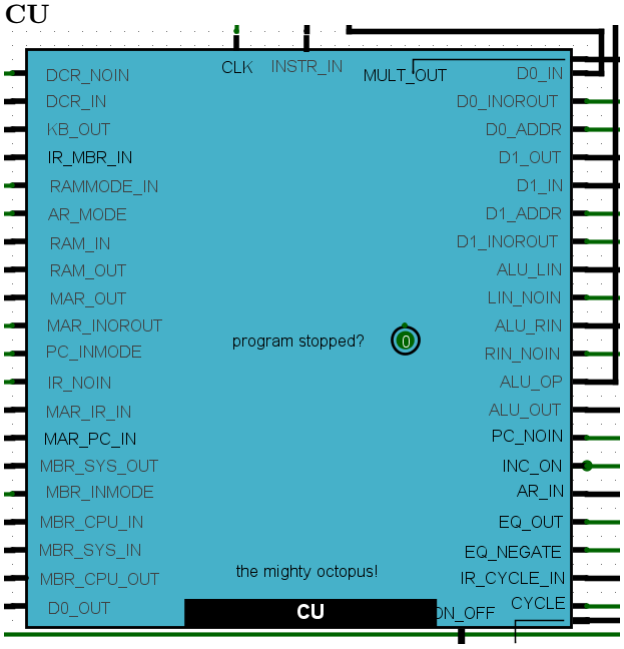
The MULT and EQ circuits complement the ALU in order to parse the bonus MULT instruction and BEQ and BNEQ.

General Purpose Registers



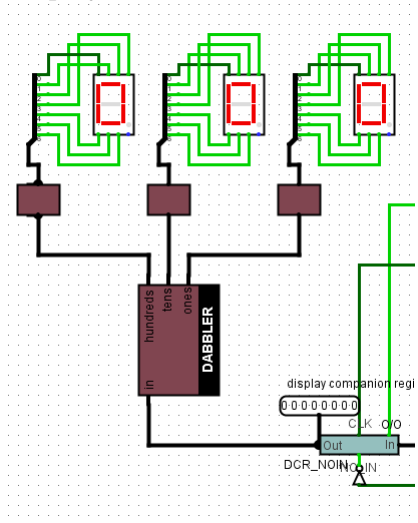
The General purpose registers receive input from either the MBR or the ALU. They have in or out modes and address inputs, which allows the CU to ask for either of the registers in particular.

CU



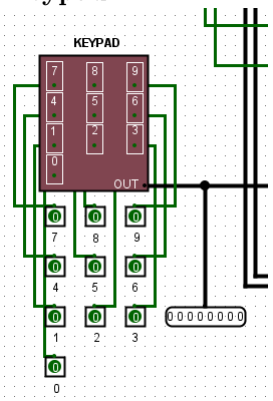
The CU receives an op code and operands from the IR and opens up the required gates for the instruction. It has a counter itself to go through the respective micro-instructions of each instruction. The CU is connected to almost every gate and input/output pins that control data flow.

Display



The display has a companion register that receives input from the general purpose data registers through the bus and has its input controlled by the CU. The display uses the Double Dabble algorithm in order to convert a byte into BCD (Binary-coded decimal) in order to display a byte as 3 decimal numbers. The corresponding BCD numbers are linked to 7-segment displays that map to the corresponding decimal numbers.

Keypad



The keypad has a companion register that receives the button pressed as a binary number. The number is sent to a specified general purpose register through the bus and the MBR when requested by the CU.

Programs

For information on how to get programs running, read the included README.txt.

2.1 Required Program

What this program does:
Multiplies (using adder in a loop) a number from RAM and a number from input buffer (keypad), answer displayed on display
Assume 1111 is the RAM value that stores the number we want to multiply by

RAM Address	RAM contents	Description
0000	0111 1000	# Load input into D1
0001	0101 0111	# If D0 isn't 0, go to prog loop
0010	1000 0000	# STOP, ends execution of program
0011	0000 0000	# Will store counter the we'll increment
0100	0000 0000	# Where we'll store answer
0101	0000 0110	# Number we want to multiply by, 6 as example
0110	0000 0001	# 1 for incrementing

Prog:

RAM Address	RAM contents	Description
0111	1010 0100	# Load current answer into D0
1000	1011 0101	# Load number we want to multiply by into D1
1001	0010 0000	# D0 = D0 + D1
1010	0110 0000	# Prints current answer
1011	1110 0100	# Store D0 into current answer
1100	1010 0011	# Load counter into D0
1101	1011 0110	# Load one into D1
1110	0010 0000	# D0 = D0 + D1, increment counter
1111	1110 0011	# Store D0 into counter

Rely on natural looping of CPU

2.2 Modified Required Program

Does the exact same thing as the required program, but uses the bonus MULT instruction.

RAM Address	RAM contents	Description
0000	0111 1000	# Load input into D1
0001	1001 1110	# D1 = D1 × constant(6)
0010	0110 1000	# Print D1
0011	1000 0000	# Stop

2.3 Countdown

Counts down from 255 to 1, showing the number on the display after every step.

RAM Address	RAM contents	Description
0000	1010 1000	# Load current number into D0
0001	0110 0000	# Print D0
0010	1011 1001	# Load 1 into D1
0011	0100 0111	# BEQ, STOP if current number is 1
0100	0011 0000	# D0 = D0 − D1
0101	1011 1010	# Load 0 into D1
0110	0101 0001	# BNEQ, jump to beginning
0111	1000 0000	# STOP
1000	1111 1111	# Current number
1001	0000 0001	# One
1010	0000 0000	# Zero

Description of Instructions

1. (CYCLE)

000
Op-Code
0-2

Not a real instruction (in the sense that it isn't a part of any program), it's what the CU uses to prepare the next instruction in RAM

2. LOAD: LD

101	x	xxxx
Op-Code	Register	Address
0-2	3	4-7

Loads the contents at an address(4-7) of RAM into a selected data register(3). The address is sent to the MAR, which is then sent to the AR. The MODE is set to read mode and the contents located at the specified address are sent to the DR, then to the MBR. Then, the contents of the MBR are sent to the specified register.

3. STORE: STR

111	x	xxxx
Op-Code	Register	Address
0-2	3	4-7

Stores the contents of a selected data register(3) into an address(4-7) of RAM. The contents of the specified register are sent to the MBR. The address is sent to the MAR, which is then sent to the AR. The MODE is set to write mode and the contents of the MBR are sent to the DR, which then sends the contents to the byte of RAM at the specified address.

4. ADD: ADD

0010	x
Op-Code	Register to store into
0-3	4

Adds the 2 data registers together and stores it into a register of choice (4). The contents of D0 are sent to the ALU through the bus. The contents of D1 are also sent to the ALU through the bus. For both of these, the CU activates the output mode on D0 and D1. Then, the ALU is set to add mode and the addition is done. Then, the result is sent to D0 through the bus and by activating the input mode on D0.

5. SUBTRACT: SUB

0011	x
Op-Code	Register to store into
0-3	4

SStores the result of subtracting the unspecified register from the specified register(4) into the specified register. Subtracts The contents of D0 are sent to the ALU through the bus. The contents of D1 are also sent to the ALU through the bus. For both of these, the CU activates the output mode on D0 and D1. Then, the ALU is set to subtract mode and the addition is done. Then, the result is sent to D0 through the bus and by activating the input mode on D0.

6. Branch equal: BEQ

0100	x
Op-Code	Address
0-3	4-7

Checks whether both data registers are equal, if they are, jumps to the specified address (4-7). The contents of D0 and D1 are directly sent to a comparison checker. When they are equal, it outputs TRUE, and this output is sent to the PC. When/if it receives this TRUE signal, it will load the adress specified.

7. Branc not equal: BNQ

0101	x
Op-Code	Address
0-3	4-7

Checks whehter both data registers aren't equal, if they aren't jumps to specified address (4-7). The contents of D0 and D1 are directly sent to a comparison checker. When they are not equal, it outputs TRUE, and this output is sent to the PC. When/if it receives this TRUE signal, it will load the adress specified.

8. PRINT: PRT

0110	x
Op-Code	Register to display
0-3	4

Prints the decimal value of the specified data register (4) to the display. The contents of the specified register are sent to the MBR through the bus, which are then sent to the Display Companion Register through the bus. Then, the display will show whatever is stored in the register.

9. INPUT: INP

0111	x
Op-Code	Register to store into
0-3	4

Loads the value of the button pressed on the keyboard into the specified data register(4). The input is converted to binary and sent to the MBR through the bus, which is then sent to the specified register through the bus.

10. STOP: STOP

1000
Op-Code
0-3

Permanently activates a pin in the CPU, which prevents the CYCLE command from being executed and therefore stops the program.

11. MULTIPLICATION: MULT

1001	x	xxx
Op-Code	Register to store into	Constant
0-3	4	5-7

Multiplies the specified data register (4) by a constant (5-7) and stores it in the original register. The contents of the specified register are sent to the left temporary register of the ALU. The instruction is sent to the CU, which isolates the constant stored within the instruction. Both of these are sent to the MULT calculator, which performs the operation and sends the output to the MBR, which then gets sent back to the specified register.