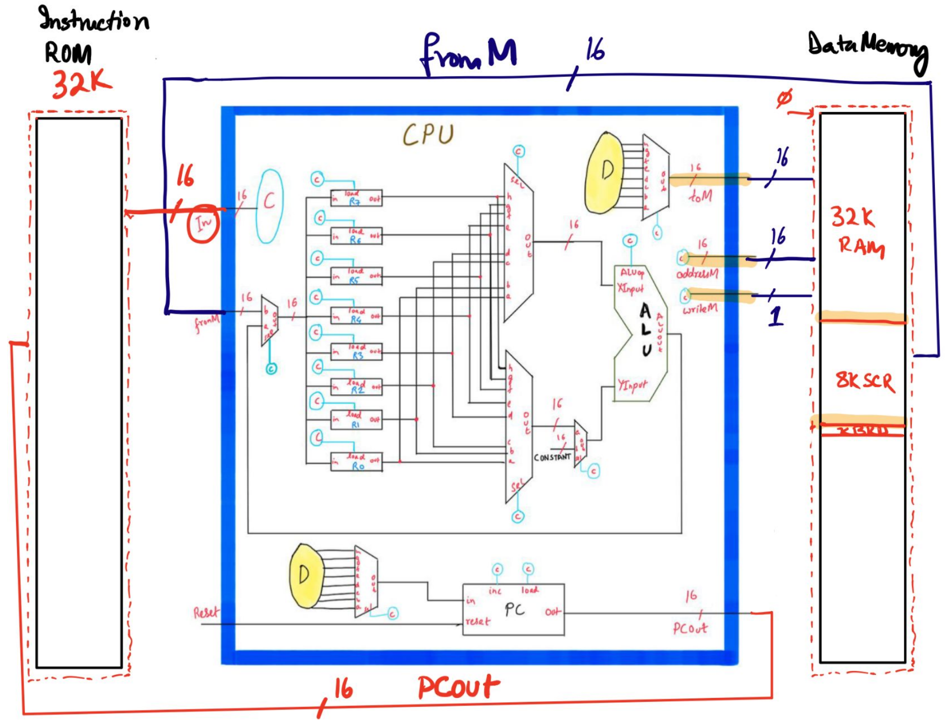
BITBOT Architecture

CSCE 312: Computer Organization

Overview

- While HACK is a great introduction to assembly language, it does not reflect popular assembly languages used in 2023
- Introducing BITBOT, a simplified RISC CPU architecture modelled heavily after MIPS/ARM
- BITBOT computer contains 3 parts:
    - ROM32K
    - CPU
    - Data Memory

Overview



Instruction ROM
32K
fromM
16
Data Memory
CPU
ø
32K RAM
8K SCR
KBRD
In
C
R7
R6
R5
R4
R3
R2
R1
R0
load
in
out
ALU
ALUop
XInput
YInput
toM
addressM
writeM
1
CONSTANT
PC
inc
load
reset
Reset
PCout
16
PCout

ROM32K

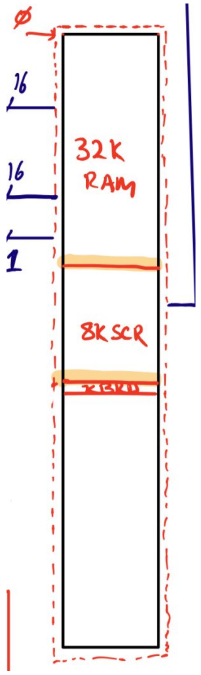
- BUILT-IN chip
- No need to do anything beside connecting it to the computer

CPU

- Extension of TOY CPU to support a wider range of instructions
- 16-bit architecture (i.e. a word is defined as 16 bits)
- 8 General Purpose Registers, 16-bit

Data Memory

- 40K+1 16-bit data words
- Divided into 3 regions
  - 32K words for holding data (RAM)
  - 8K words for memory-mapped screen output
  - 1 word for memory-mapped keyboard input device



Computer

- Combine 3 parts together
- Only for wiring parts, no additional logic required

Further details

- The recommended order of implementation:
  - Data Memory
  - CPU
  - Computer

Further details: Memory

- Built-in chips: RAM32K(in=, load=, address=, out=), Screen(in=, load=, address=, out=), Keyboard(out=)
  - RAM32K: only bit 14 to 0 are used (2^15 = 32K), thus must check address[15] = 0 and load = 1 before accessing the RAM
  - Screen: use address[13..15] for loading (load if it is 100), address[0..12] is for the mapped memory positions on the screen
  - Keyboard: should be self-explanatory
  - Use a Mux8Way16 to glue everything together, use the hint above to figure out the correct selector bits

Further details: CPU

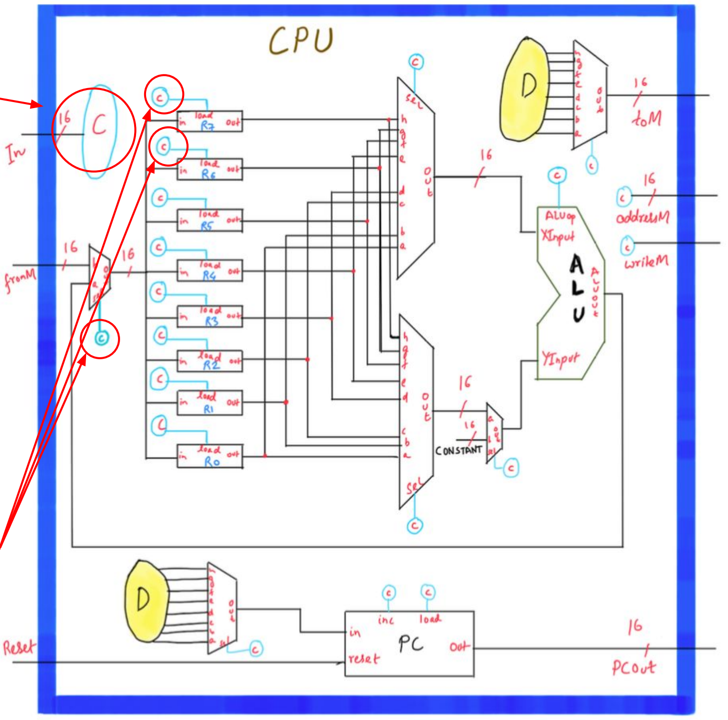
- Schematic mostly given for datapath
- Recommended strategy for controlpath:
  - Go through each instruction (start with a set of instructions)
  - Register control (whenever register is written to)
  - Arithmetic (ALU)
  - Memory read/write (addressM, writeM, toM, and Screen)
  - Branching (control the PC)

Further details: CPU – example

- Let’s look at designing the controlpath for register control

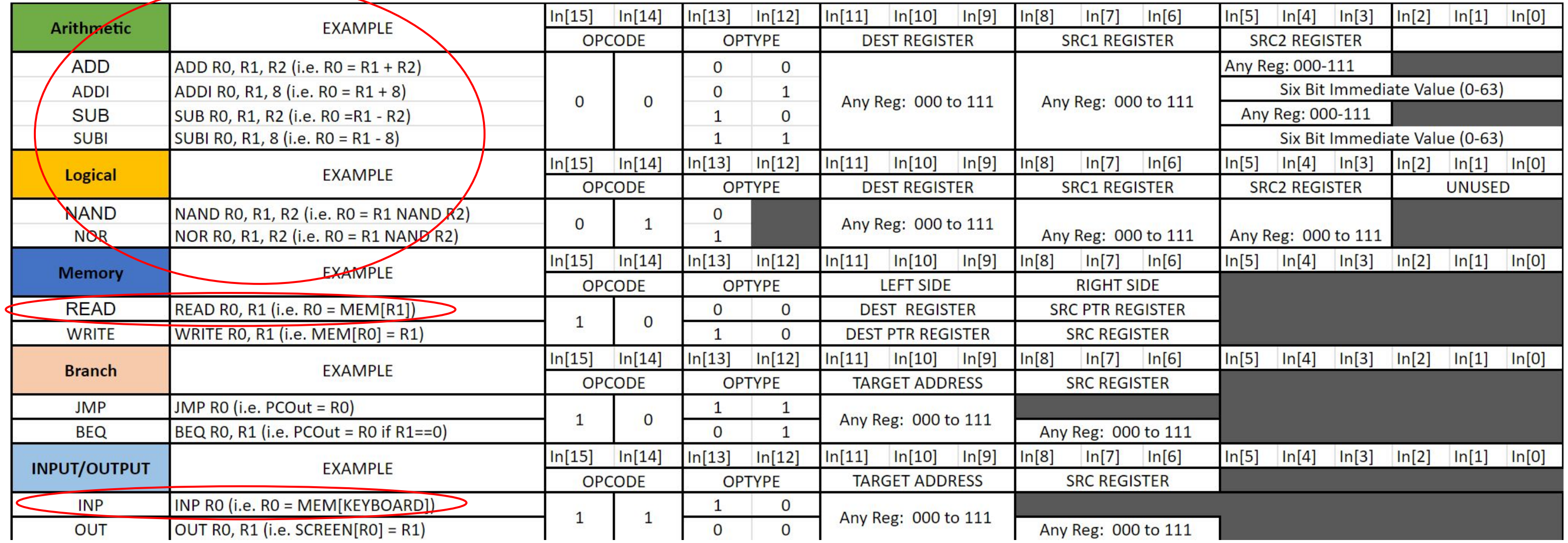
Further details: CPU – example

Controlpath source



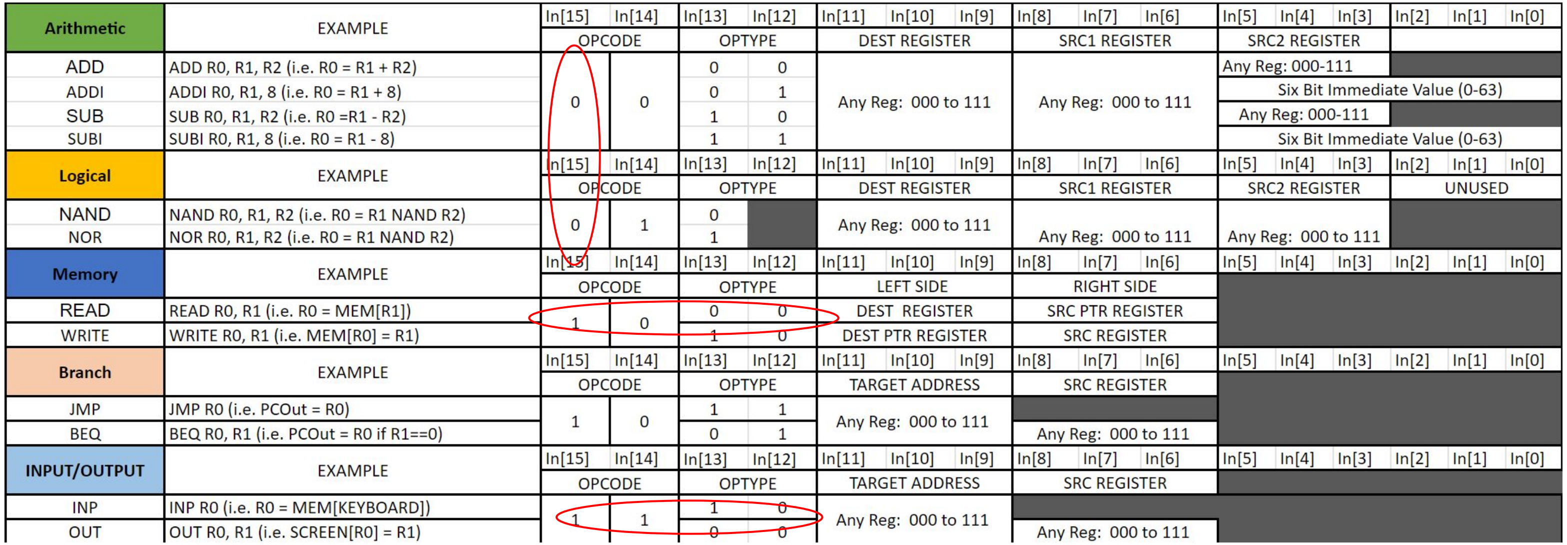
Controlpath related to register control

Further details: CPU – example



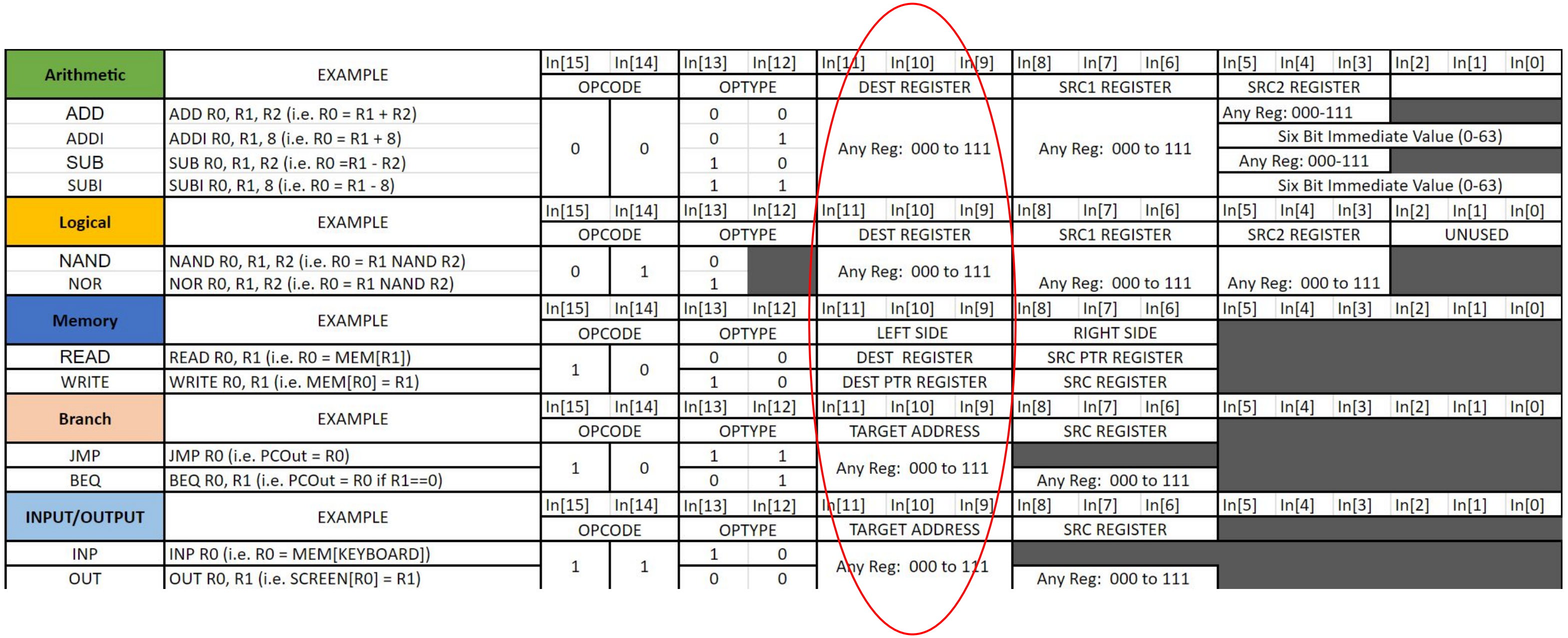
Arithmetic	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		DEST REGISTER			SRC1 REGISTER			SRC2 REGISTER					
ADD	ADD R0, R1, R2 (i.e. R0 = R1 + R2)	0	0	0	0	Any Reg: 000 to 111			Any Reg: 000 to 111			Any Reg: 000-111			▨		
ADDI	ADDI R0, R1, 8 (i.e. R0 = R1 + 8)			0	1							Six Bit Immediate Value (0-63)					
SUB	SUB R0, R1, R2 (i.e. R0 =R1 - R2)			1	0							Any Reg: 000-111			▨		
SUBI	SUBI R0, R1, 8 (i.e. R0 = R1 - 8)			1	1							Six Bit Immediate Value (0-63)					
Logical	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		DEST REGISTER			SRC1 REGISTER			SRC2 REGISTER			UNUSED		
NAND	NAND R0, R1, R2 (i.e. R0 = R1 NAND R2)	0	1	0	▨	Any Reg: 000 to 111			Any Reg: 000 to 111			Any Reg: 000 to 111			▨		
NOR	NOR R0, R1, R2 (i.e. R0 = R1 NAND R2)			1	▨												
Memory	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		LEFT SIDE			RIGHT SIDE			▨					
READ	READ R0, R1 (i.e. R0 = MEM[R1])	1	0	0	0	DEST REGISTER			SRC PTR REGISTER			▨					
WRITE	WRITE R0, R1 (i.e. MEM[R0] = R1)			1	0	DEST PTR REGISTER			SRC REGISTER			▨					
Branch	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		TARGET ADDRESS			SRC REGISTER			▨					
JMP	JMP R0 (i.e. PCOut = R0)	1	0	1	1	Any Reg: 000 to 111			▨			▨					
BEQ	BEQ R0, R1 (i.e. PCOut = R0 if R1==0)			0	1				Any Reg: 000 to 111			▨					
INPUT/OUTPUT	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		TARGET ADDRESS			SRC REGISTER			▨					
INP	INP R0 (i.e. R0 = MEM[KEYBOARD])	1	1	1	0	Any Reg: 000 to 111			▨			▨					
OUT	OUT R0, R1 (i.e. SCREEN[R0] = R1)			0	0				Any Reg: 000 to 111			▨					

Further details: CPU – example



Arithmetic	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		DEST REGISTER			SRC1 REGISTER			SRC2 REGISTER					
ADD	ADD R0, R1, R2 (i.e. R0 = R1 + R2)	0	0	0	0	Any Reg: 000 to 111			Any Reg: 000 to 111			Any Reg: 000-111			▨	▨	▨
ADDI	ADDI R0, R1, 8 (i.e. R0 = R1 + 8)			0	1							Six Bit Immediate Value (0-63)					
SUB	SUB R0, R1, R2 (i.e. R0 =R1 - R2)			1	0							Any Reg: 000-111			▨	▨	▨
SUBI	SUBI R0, R1, 8 (i.e. R0 = R1 - 8)			1	1							Six Bit Immediate Value (0-63)					
Logical	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		DEST REGISTER			SRC1 REGISTER			SRC2 REGISTER			UNUSED		
NAND	NAND R0, R1, R2 (i.e. R0 = R1 NAND R2)	0	1	0	▨	Any Reg: 000 to 111			Any Reg: 000 to 111			Any Reg: 000 to 111			▨	▨	▨
NOR	NOR R0, R1, R2 (i.e. R0 = R1 NAND R2)			1	▨										▨	▨	▨
Memory	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		LEFT SIDE			RIGHT SIDE			▨	▨	▨	▨	▨	▨
READ	READ R0, R1 (i.e. R0 = MEM[R1])	1	0	0	0	DEST REGISTER			SRC PTR REGISTER			▨	▨	▨	▨	▨	▨
WRITE	WRITE R0, R1 (i.e. MEM[R0] = R1)			1	0	DEST PTR REGISTER			SRC REGISTER			▨	▨	▨	▨	▨	▨
Branch	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		TARGET ADDRESS			SRC REGISTER			▨	▨	▨	▨	▨	▨
JMP	JMP R0 (i.e. PCOut = R0)	1	0	1	1	Any Reg: 000 to 111			▨	▨	▨	▨	▨	▨	▨	▨	▨
BEQ	BEQ R0, R1 (i.e. PCOut = R0 if R1==0)			0	1				Any Reg: 000 to 111			▨	▨	▨	▨	▨	▨
INPUT/OUTPUT	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		TARGET ADDRESS			SRC REGISTER			▨	▨	▨	▨	▨	▨
INP	INP R0 (i.e. R0 = MEM[KEYBOARD])	1	1	1	0	Any Reg: 000 to 111			▨	▨	▨	▨	▨	▨	▨	▨	▨
OUT	OUT R0, R1 (i.e. SCREEN[R0] = R1)			0	0				Any Reg: 000 to 111			▨	▨	▨	▨	▨	▨

Further details: CPU – example



Arithmetic	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		DEST REGISTER			SRC1 REGISTER			SRC2 REGISTER					
ADD	ADD R0, R1, R2 (i.e. R0 = R1 + R2)	0	0	0	0	Any Reg: 000 to 111			Any Reg: 000 to 111			Any Reg: 000-111			▨	▨	▨
ADDI	ADDI R0, R1, 8 (i.e. R0 = R1 + 8)			0	1							Six Bit Immediate Value (0-63)					
SUB	SUB R0, R1, R2 (i.e. R0 =R1 - R2)			1	0							Any Reg: 000-111			▨	▨	▨
SUBI	SUBI R0, R1, 8 (i.e. R0 = R1 - 8)			1	1							Six Bit Immediate Value (0-63)					
Logical	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		DEST REGISTER			SRC1 REGISTER			SRC2 REGISTER			UNUSED		
NAND	NAND R0, R1, R2 (i.e. R0 = R1 NAND R2)	0	1	0	▨	Any Reg: 000 to 111			Any Reg: 000 to 111			Any Reg: 000 to 111			▨	▨	▨
NOR	NOR R0, R1, R2 (i.e. R0 = R1 NAND R2)			1	▨										▨	▨	▨
Memory	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		LEFT SIDE			RIGHT SIDE			▨	▨	▨	▨	▨	▨
READ	READ R0, R1 (i.e. R0 = MEM[R1])	1	0	0	0	DEST REGISTER			SRC PTR REGISTER			▨	▨	▨	▨	▨	▨
WRITE	WRITE R0, R1 (i.e. MEM[R0] = R1)			1	0	DEST PTR REGISTER			SRC REGISTER			▨	▨	▨	▨	▨	▨
Branch	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		TARGET ADDRESS			SRC REGISTER			▨	▨	▨	▨	▨	▨
JMP	JMP R0 (i.e. PCOut = R0)	1	0	1	1	Any Reg: 000 to 111			▨	▨	▨	▨	▨	▨	▨	▨	▨
BEQ	BEQ R0, R1 (i.e. PCOut = R0 if R1==0)			0	1				Any Reg: 000 to 111			▨	▨	▨	▨	▨	▨
INPUT/OUTPUT	EXAMPLE	In[15]	In[14]	In[13]	In[12]	In[11]	In[10]	In[9]	In[8]	In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]
		OPCODE		OPTYPE		TARGET ADDRESS			SRC REGISTER			▨	▨	▨	▨	▨	▨
INP	INP R0 (i.e. R0 = MEM[KEYBOARD])	1	1	1	0	Any Reg: 000 to 111			▨	▨	▨	▨	▨	▨	▨	▨	▨
OUT	OUT R0, R1 (i.e. SCREEN[R0] = R1)			0	0				Any Reg: 000 to 111			▨	▨	▨	▨	▨	▨

Further details: CPU

When design controlpath, ALWAYS look at the register sheet

Further details: Computer

- Instantiate the ROM32K
- That’s it :)

Where to find additional information

- Provided resources
- The TOY CPU design presentations
- The HACK CPU lectures

Next week

Verifying the whole system + work week