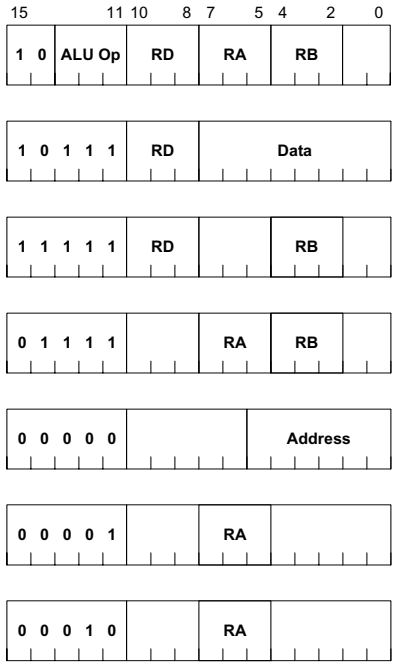
x370 Processor Definition

The x370 processor is a simple 16-bit architecture based on the ALU and register file you have designed for homework. The x370 has room for 8 registers and separate instruction and data memories. The instruction memory has 64 16-bit instructions and data memory has 256 16-bit data values. All x370 instructions can be executed in a single cycle. The instruction set has been designed so that it can be extended for those who think it should do more. The instruction memory can also be expanded easily to 256 instructions.

Instruction Set



RD = RA op RB — ALU instruction

RD = Data — LDI - Load Immediate

RD = Dmem[RB] — LDR - Load Register

Dmem[RB] = RA — STR - Store Register

PC = Address — BR - Branch

if (RA==0) PC = PC+2 — SKZ - Skip on Zero

if (RA<0) PC = PC+2 — SKN - Skip on Negative

ALU Instructions

Name	Op code	Operation	Comments
ADD	10000	RD ← RA + RB	
XOR	10001	RD ← RA ⊕ RB	
INC	10010	RD ← RA + 1	
PASSA	10011	RD ← RA	
Reserved	10100		Available for new ALU operation
Reserved	10101		Available for new ALU operation
SUB	10110	RD ← RB – RA	Note order of operands
LDI	10111	RD ← Data (sign extended to 16 bits)	Non-ALU instruction: Load immediate loads data from instruction

Branch and Skip Instructions

The branch and skip instructions allow the program to execute loops and execute different instructions depending on the result of an ALU operation. Branch is unconditional, while skips test the value in a register. Skips are typically followed by a branch instruction, for example, to return to the top of a loop.

Name	Op code	Operation	Comments
BR	00000	PC ← Address	Unconditional branch
SKZ	00001	if (RA==0) PC ← PC + 2; else PC ← PC + 1	
SKN	00010	if (RA<0) PC ← PC + 2; else PC ← PC + 1	

Load/Store Instructions

Data memory is accessed via the load and store instructions, which transfer a single value between a register and a location in data memory, whose address is given in register RB. Only the low-order 8 bits of RB is used for the address since the data memory has 256 locations.

Name	Op code	Operation	Comments
LDR	11111	RD ← DMEM[RB]	
STR	01111	DMEM[RB] ← RA	

Implementing the x370 Processor

We will implement the x370 in several steps instead of trying to do it all at once. This way, you can make sure it works as you go along.

x370 Model 0 – ALU Instructions

The base processor is very simple – it executes ALU instructions only, starting after reset with the instruction at address 0, and then executing instructions at 1, 2, etc. You have already implemented this in homeowork. Your job is to add instructions and features to implement the full processor.

x370 Model 1 – Load Immediate and Branch Instructions (hand in May 28)

Implement the load immediate (LDI) instruction, which allows the program to load a constant that is part of the instruction into a register. This 8-bit constant is sign extended to allow negative constants.

The Model 1 also incorporates a branch instruction and two skip instructions, which allows a program to execute loops and to branch based on the results of an ALU operation. The branch (BR) instruction specifies the address of the next instruction to execute. The two skip instructions allow the program to execute two different instructions depending on the contents of a register. The skip on zero (SKZ) instruction skips the next instruction if the specified register is zero. The skip on negative (SKN) instruction skips the next instruction if the specified register is less than zero.

x370 Model 2 – Load/Store and Data Memory (hand in electronically by June 4)

So far, all the data used by the program is kept in the registers in the register file. To solve interesting problems, we need to have memory which contains input data and output data, as well as temporary data as needed. The Model 3 has a separate data memory with 256 locations. This memory is accessed via the LDR, Load Register, and STR, Store Register, instructions. In both cases, the address of the location in data memory is given by register RB. The LDR instruction loads this memory location into RD, and the STR instruction stores RA to this memory location.

The data memory is implemented using the dram.v module. This module has a parameter that gives the name of the file that is used to initialize the memory contents. You can change the file name by right-clicking on the dram module and changing this parameter.

We will run several benchmark programs on your processor to determine whether it works.

Extra Credit Program (hand in electronically by June 4)

Write a program that runs on the x370 that does a "phone book search". We will give you a list of 120 pairs of numbers, each representing a name and a phone number pair. These will be in a "dram.dat" file, with the first pair of numbers at addresses 10 and 11. Your program will then search for a "name" in this set of data, and return the corresponding "phone number". We will use location 0 of the data memory as the "name" to search for. Your program should store the associated phone number in data memory location 1 and then halt by branching and looping at location 255 (the last instruction). You may use the first 10 locations in memory however you like. You must turn in your project electronically and we will run your program to test it.

You may consider executing a simple linear search or a faster search method. You may also consider extending the instruction set with instructions that make the program shorter and faster. If so, you must maintain the base instruction set that it runs our benchmark programs.