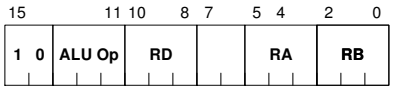
x370 Processor Definition

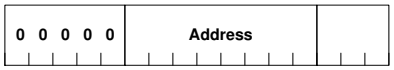
The x370 processor is a simple 16-bit architecture based on the ALU and register file that is designed in class. The x370 is a Load/Store architecture and has 8 registers and separate instruction and data memories. The instruction memory has up to 256 16-bit instructions and data memory has up to 256 16-bit data values. Simpler versions of the x370 can be constructed using fewer registers, smaller memories and fewer instructions. All x370 instructions can be executed in a single cycle.

Instruction Set

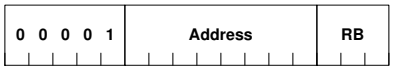


RD = RA op RB — ALU instruction

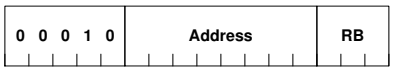
RD = Data — LDI - Load Immediate



PC = Address — BR - Branch



if (RB==0) PC = Address — BRZ - Branch on Zero



if (RB<0) PC = Address — BRN - Branch on Negative

RD = Dmem[RB] — LDR - Load Register

Dmem[RB] = RA — STR - Store Register

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
XNOR	10101	RD ← ¬(RA ⊕ RB)	
SUB	10110	RD ← RB – RA	Note order of operands
LDI	10111	RD ← Data (sign extended to 16 bits)	Non-ALU instruction: Load immediate data from instruction

Branch Instructions

Branch instructions allow the program to execute loops and execute different instructions depending on the result of an ALU operation. The conditional branch instructions test the value in register RB. Branches are typically executed right after the ALU instruction that generates the value to be tested.

Name	Op code	Operation	Comments
BR	00000	PC ← Address	Unconditional branch
BRZ	00001	if (RB == 0) PC ← Address; else PC ← PC + 1	Branch if Zero
BRN	00010	if (RB < 0) PC ← Address; else PC ← PC + 1	Branch if Negative

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 at most 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.

Model 0 – Register store (hand in Friday, Nov. 19)

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 homework. Your job is to construct a register store to add instructions and features to implement the full processor.

x370 Model 1 – Load Immediate and Branch Instructions (complete by Friday, Dec. 3)

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

The model 2 incorporates the branch instructions, which allows a program to execute loops and to branch based on the value of a register. There are three branch instructions: BR, unconditional branch, BZ, branch if the register RB is 0, and BN, branch if the register RB negative. The branch instructions specify the address of the next instruction to execute if the branch condition holds.

x370 Model 3 – Load/Store and Data Memory (complete by Wednesday Dec. 8)

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.

Extra Credit (hand in electronically, Friday Dec. 10)

You will 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. You may save this in the data memory however you like by defining your own "dram.dat" initial data file. 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 63 (the last instruction). You must turn in your project electronically and we will run your program to test it.