Raul Aguilar

CS 220

October 13, 2019

Homework 6

CHIP Memory {

IN in[16], load, address[15];

OUT out[16];

PARTS:

// Put your code here:

DMux4Way(in=load, sel=address[13..14], a= rama, b=ramb, c=screen, d=key);

Or(a=rama, b=ramb, out=ram);

RAM16K(in=in, address=address[0..13], load=ram, out=rout);

Screen(in=in, address=address[0..12], load=screen, out=sout);

Keyboard(out=kout);

Mux4Way16(a=rout, b=rout, c=sout, d=kout, sel=address[13..14], out=out);

}

A screenshot of a social media post

Description automatically generated

CHIP Computer {

IN reset;

PARTS:

// Put your code here:

ROM32K(address=pc, out=instruction);

CPU(inM=data, instruction=instruction, reset=reset, writeM=load, outM=value, addressM=address, pc=pc);

Memory(address=address, in=value, load=load, out=data);

}

ComputerAdd.tst

A screenshot of a social media post

Description automatically generated

ComputerMax.tst

A screenshot of a social media post

Description automatically generated

ComputerRect.tstA screenshot of a social media post

Description automatically generated

CHIP CPU {

IN inM[16], // M value input (M = contents of RAM[A])

instruction[16], // Instruction for execution

reset; // Signals whether to re-start the current

// program (reset==1) or continue executing

// the current program (reset==0).

OUT outM[16], // M value output

writeM, // Write to M?

addressM[15], // Address in data memory (of M)

pc[15]; // address of next instruction

PARTS:

// Put your code here:

// Determine if A-instruction or C-instruction

Not(in=instruction[15], out=nin);

Mux16(a=aluout, b=instruction, sel=nin, out=inA);

// Store into A register

Or(a=nin, b=instruction[5], out=regA);

ARegister(in=inA, load=regA, out=aout, out[0..14]=addressM);

// Determine if comp is from A-register or M input

And(a=instruction[15], b=instruction[12], out=AM);

Mux16(a=aout, b=inM, sel=AM, out=AMout);

ALU(x=D, y=AMout, zx=instruction[11], nx=instruction[10], zy=instruction[9], ny=instruction[8], f=instruction[7], no=instruction[6], out=aluout, out=outM, zr=zr, ng=ng);

// Determine if store into D-register

And(a=instruction[15], b=instruction[4], out=regD);

DRegister(in=aluout, load=regD, out=D);

// Or write to address in Memory

And(a=instruction[15], b=instruction[3], out=writeM);

// Determine jumps

And(a=instruction[15], b=instruction[0], out=jgt);

And(a=instruction[15], b=instruction[1], out=jeq);

And(a=instruction[15], b=instruction[2], out=jlt);

Not(in=ng, out=pos);

Not(in=zr, out=nzr);

And(a=pos, b=nzr, out=posnzr);

And(a=posnzr, b=jgt, out=gt);

And(a=zr, b=jeq, out=eq);

And(a=ng, b=jlt, out=lt);

Or(a=gt, b=eq, out=geq);

Or(a=geq, b=lt, out=jmp);

// if there is a jump, set PC to A

// else set PC to PC+1

PC(in=aout, load=jmp, inc=true, reset=reset, out[0..14]=pc);

}

A screenshot of a social media post

Description automatically generated