TIS-100 ISA

Tobias Shapinsky
Henry Rachootin

Instruction Set[1]

NAME, MNEUMONIC		FOR-MAT	OPERATION		OPCODE (hex)
Add	add	R	$ACC = ACC + P[src]$	(1)	0_{hex}
Add Immediate	addi	I	$ACC = ACC + Imm$		5_{hex}
Subtract	sub	R	$ACC = ACC - P[src]$	(1)	1_{hex}
Subtract Immediate	subi	I	$ACC = ACC - Imm$		6_{hex}
Jump Relative Offset	jro	R	$PCnext = PC + P[src]$	(1)	2_{hex}
Jump Relative Offset Immediate	jroi	I	$PCnext = PC + Imm$		7_{hex}
Move	mov	R	$P[dst] = P[src]$	(1,3)	3_{hex}
Move Immediate	movi	I	$P[dst] = Imm$		4_{hex}
Jump	jmp	J	$PCnext = Label$		b_{hex}
Jump Not Zero	jnz	J	$if(ACC \neq 0)PCnext = Label$		d_{hex}
Jump Equal Zero	jez	J	$if(ACC = 0)PCnext = Label$		c_{hex}
Jump Less Than Zero	jlz	J	$if(ACC < 0)PCnext = Label$		f_{hex}
Jump Greater Than Zero	jgz	J	$if(ACC > 0)PCnect = Label$		e_{hex}
Swap	swp	R	$ACC = BAK$	(2)	8_{hex}
Save	sav	R	$BAK = ACC$		9_{hex}
Negate	neg	R	$ACC = -ACC$		a_{hex}

(1) Blocks until port is populated with value
(2) Happens simultaneously s.t. ACC and BAK are swapped
(3) Writing to any dst other than ACC or NIL will take at least 2 cycles

Modules

Registers

Each TIS-100 module comes equipped with three 11-bit registers (ACC, BAK, NIL). The only register that can be read or written to is ACC. However, the values of ACC and BAK may be swapped at any time through the use of the SWP instruction. The third register (NIL) always reads the value zero and may be substituted for any src or dst port.

Instructions

The TIS-100 supports 16 different instructions (detailed above) which each consist of a 4-bit opcode followed by 14-bits of data determined by the instruction's type (figure 1).
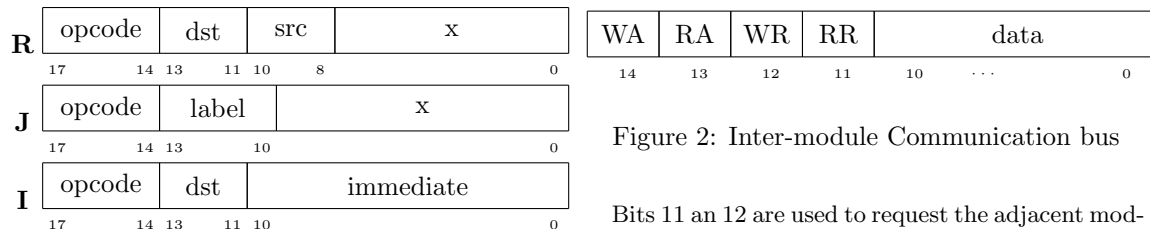
Each TIS-100 module has 270 bits of program memory which is enough to store up to 15 18-bit instructions. It should be noted that even though there is no specific NOP instruction a instruction consisting of 18-bits of zeros will be decoded as ADD NIL and thus functions as a NOP.

Ports

Even though each TIS-100 module is very powerful on its own, the real power comes with the ability to connect TIS-100 modules to each other. For this reason each TIS-100 module has four directional ports (UP, DOWN, LEFT, RIGHT) which may be used to communicate with neighboring modules via a 15-bit inter-module communication bus (figure 2).

Figure 1: Instruction formats

Figure 2: Inter-module Communication bus

Bits 11 an 12 are used to request the adjacent module to write or read from the bus respectively. Bits 13 and 14 are used to signal to the requesting port

---

[1]TIS-100 Manual http://www.zachtronics.com/images/TIS-100P%20Reference%20Manual.pdf

that the write or read has been completed successfully. The remaining bits are reserved for the data being transmitted between modules.

It should be noted that a write operation will always take at least two CPU cycles.

Pseudo-Ports

On top of the four directional ports each TIS-100 module is also equipped with two pseudo-ports (LAST, ANY). When writing to or reading from, the LAST pseudo-port acts as an alias for the last port that was used. In the case that no port has be written to or read from yet the LAST pseudo-port acts as the NIL register.

The behaviour for the ANY pseudo-port is a little more complicated. When reading from ANY the CPU looks for any neighboring module with a high READ_REQ flag, if there are multiple modules it uses the order (LEFT, RIGHT, UP, DOWN), to determine precedence and then reads its value and ACKs the module. If there are no neighboring modules with a high READ_REQ flag it instead broadcasts its desire to read by setting the WRITE_REQ flag high on all of its directional ports and waits for a module to signal that it has something to send.

In the case of writing to ANY the CPU looks for any neighboring modules with a high WRITE_REQ flag, if there are none it blocks for a cycle and checks again. In the case that there are multiple neighboring modules with a high WRITE_REQ flag the CPU uses the order (UP, LEFT, RIGHT, DOWN), to determine precedence and then writes its value and ACKs the module.

Execution

The TIS-100 is a three-phase multi-cycle CPU. In the first phase (the read phase) each module decodes the instruction pointed to by the program counter (PC). At this point if the instruction required reading from a directional port (for pseudo-ports see section on pseudo-ports) the CPU would check the requisite port for a high READ_REQ flag, if none is found it transitions to the READ mode, set its WRITE_REQ flag to high and blocks for a cycle. Once the target module has a high READ_REQ flag it reads the data and transitions back into READ mode.

In the execution phase, if the CPU is still in RUN mode it will execute the decoded instruction. If the mode is WRITE and the target is ANY the CPU will update the target if any of its inter-module communication buses has a high WRITE_REQ flag.

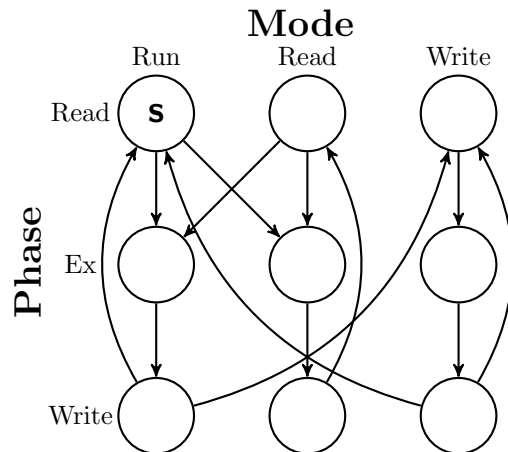
Figure 3: Internal CPU state machine

In the write phase, if the instruction requires a write to a port the CPU will set its READ_REQ flag to high on the inter-module communication bus of the corresponding port and transition to WRITE mode. If the target of the write operation is a register (ACC or NIL) it will immediately write to the register without changing modes.

Stack Memory

In addition to other TIS-100 modules the TIS-100 may also be connected to a stack memory module. Each stack memory module can hold up to 15 11-bit values. These values can be accessed through the inter-module communication bus using the same protocol as a normal TIS-100 module. A read will pop one value off the stack and a write will push one value onto the stack.
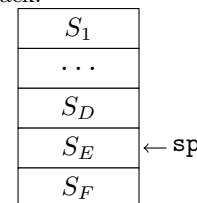

Figure 4: Stack memory

On the stack memory's inter-module communication bus the READ_REQ and WRITE_REQ flags will always be high, unless the value of the stack pointer is either zero or 15, at which point the module has either run out of memory or values in memory.