

PeANUt Module – Overview

- a simple microprocessor simulator for teaching purposes
- main topics:
 - PeANUt architecture, machine language and assembler programming
 - branches and conditions, loops, input/output, traps, macros
 - procedures and functions
 - translating C programs into assembler language
 - ◆ a low-level execution model for C; how a compiler works
 - ◆ documenting the assembly language program
- bring your Specification of the PeANUt computer (reading brick) to all lectures and tute/labs (can still buy!)
- announcements:
 - Assignment 1: C programs – motivations & clarifications; loop unrolling example: unroll.c.txt
 - no lecture Monday; Monday tute/lab group should 'sit-in' on later classes

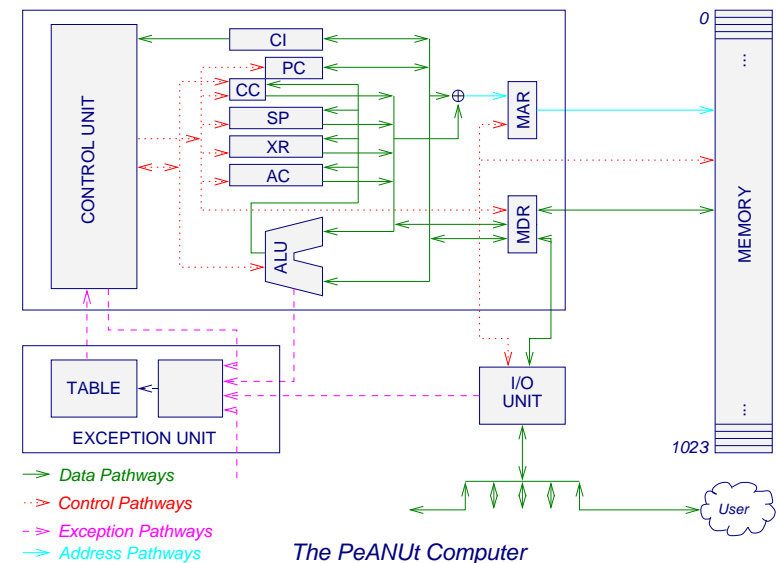
Microprocessors

- there are many well known microprocessors:
 - Intel x86 series, Pentium, Celeron, Xeon, etc.
 - AMD Opteron; Intel Itanium
 - Motorola 680xx series, PowerPC
 - SPARC / UltraSPARC
 - MIPS
 - Compaq Alpha
 - PeANUt
- we shall investigate the design and operation of the ANU illustrative microprocessor, the PeANUt

PeANUt – The Basics

- microprocessors
- PeANUt architecture
 - memory
 - CPU
 - registers
 - the execution cycle
 - instruction set
 - address modes
- reference: [PeANUt Spec, sect 1–2.7]

The PeANUt Architecture



Execution Cycle

- the control unit decodes the current instruction and controls the execution by controlling the individual components of the CPU
- execution cycle: REPEAT
 - PC \leftarrow PC + 1
 - CI \leftarrow mem[PC-1] (instruction Fetch)
 - Evaluate Operand
 - Execute Instruction
 - Service Exceptions (if any)
- FOREVER
- the instruction fetch sequence:
 - MAR \leftarrow PC-1
 - (memory) Read, Enable
 - MDR \leftarrow mem[MAR]
 - CI \leftarrow MDR
 - Control Unit decodes CI
- data flow varies with different instructions consider LOAD 20₁₀

Instruction Addressing Modes

- (Q: why can't PeANUt instructions just have a single format?)
- the mode determines how the operand specifier (opspec) is interpreted
- every arithmetic and load/store instruction has one of the following modes:
 - (subsequent diagrams indicate, for each of these, how the opspec is used)
 - 000: immediate mode
 - 001: direct mode
 - 010: indirect mode
 - 011: indexed mode (*later*)
 - 100: stack mode (*later*)
- our analogy is about ways of delivering a briefcase which may be in a bank of lockers

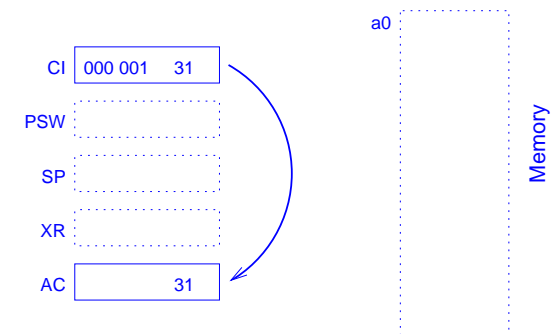
The PeANUt Instruction Set

- all instructions are 16 bits long (1 memory cell)
- each instruction has up to three components:
 - an opcode, identifying the instruction ([PeANUt Spec, Table 1])
 - e.g. 011 identifies an add instruction
 - an operand specifier (opspec)
 - an (addressing) mode (may be implicit (and fixed), or N/A)
- the opspec and mode determine the operand:
 - the data upon which the instruction operates (also may be implicit)
 - e.g. an operand of 1 may result in 1 being added to AC
- all instr'ns may be classified according to their instruction format ([PeANUt Spec, p. 3]):

format	fields												
One	<table border="1"> <tr> <td>15</td><td>13</td><td>12</td><td>10</td><td>9</td><td>0</td> </tr> <tr> <td colspan="2">mode</td> <td colspan="2">opcode</td> <td colspan="2">opspec</td> </tr> </table>	15	13	12	10	9	0	mode		opcode		opspec	
15	13	12	10	9	0								
mode		opcode		opspec									
Two	<table border="1"> <tr> <td>15</td><td>10</td><td>9</td><td>0</td> </tr> <tr> <td colspan="2">opcode</td> <td colspan="2">opspec</td> </tr> </table>	15	10	9	0	opcode		opspec					
15	10	9	0										
opcode		opspec											
Three	<table border="1"> <tr> <td>15</td><td>9</td><td>8</td><td>0</td> </tr> <tr> <td colspan="2">opcode</td> <td colspan="2">unused</td> </tr> </table>	15	9	8	0	opcode		unused					
15	9	8	0										
opcode		unused											

Immediate Mode (mode bits 000)

- suppose the opspec is X
- the operand to be used is just X



- there is no memory access involved here
- in our analogy, it corresponds to just giving the briefcase

