

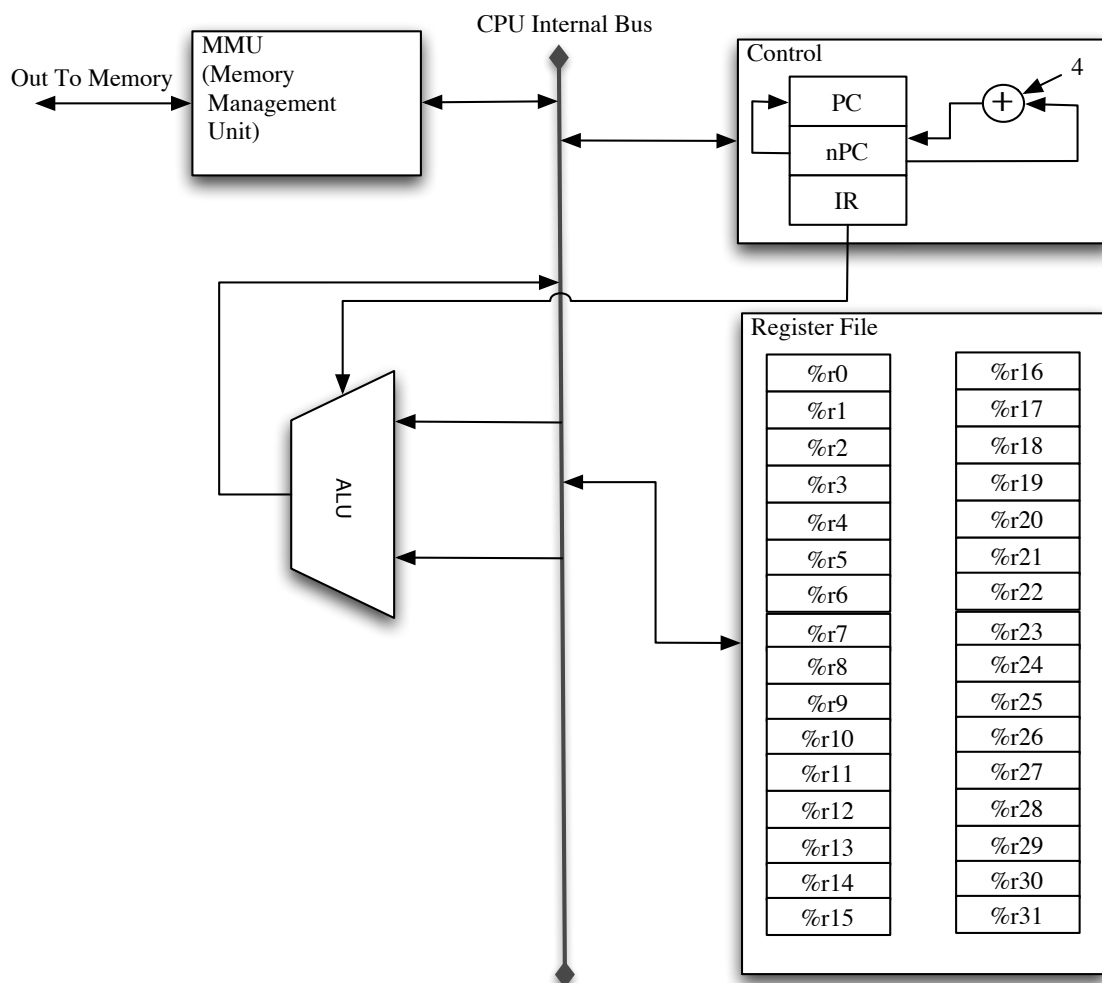
CmpSci 201

Homework 1

In this homework, you will explore the fetch-execute cycle in detail. You will do so by analyzing the flow of a series of instructions through a simplified 32-bit SPARC core.

1 A Simple SPARC

The SPARC processor is a RISC processor popular in Sun Microsystems workstations. Currently on its ninth version, the latest version of the SPARC architecture is 64 bit, but today we'll examine a 32-bit version.



The SPARC has 31 32-bit wide general purpose registers (%r0 is actually hard-wired to be the value 0), labeled %r1 - %r31. The internal bus and the memory bus is 32-bits wide. Each SPARC instruction is 32 bits large (that is to say, 4 bytes).

The SPARC has a unique feature: in addition to a program counter (PC) there is a next program counter (nPC). The PC update procedure is as follows: the nPC is copied into the PC and the nPC is incremented. For example if the PC is 0 and the nPC is 4, when the instruction at address 0 is executed the PC gets the value 4, and nPC gets the value 8. The SPARC memory is byte addressable, and because SPARC instructions are 4 bytes large, the nPC is increased by 4 to get the next instruction.

The control unit contains the PC, nPC, and IR registers. The PC stores the **address** of the current instruction to fetch and execute. nPC stores the **address** of the next instruction to fetch and execute. IR (the Instruction Register) holds the currently executing instruction.

The ALU (Arithmetic and Logic Unit) performs the various integer operations on the processor. The ALU knows what operations to perform because the circuitry interprets the bit value of the executing instruction, held in the IR (this is also known as decoding). The ALU can accept up to two inputs, and writes its output to the internal bus.

The register file can read values from the bus and store them in registers, or write register values out to the bus. For this simple example, you may assume that the register file can write two values to the bus at the same time and that the ALU can read both values off the bus without any difficulty.

For the purposes of this homework, you'll need to understand the following SPARC instructions:

- `add %rx %ry %rz`
adds the value in `%rx` to the value in `%ry` and stores the result into `%rz`. An immediate constant can be substituted for `%ry` (e.g. `add %r12, 42, %r13`).
- `sub %rx %ry %rz`
subtracts the value in `%ry` from `%rx` and stores the result in `%rz`. Like `add`, `sub` can also take a constant argument.

2 Exercises

1. If `PC = 0` and `nPC = 16`, after an instruction is executed, what will the values of `PC` and `nPC` be?
2. Many processors have a `mov` instruction that copies a value from one register into another. Using only `add` or `sub` (or both), how would you perform a move operation?
3. The following exercise refers to the code below:

```
add %r1, %r2, %r3
sub %r3, %r2, %r1
add %r2, %r2, %r3
sub %r3, %r2, %r4
```

- a) If the address of the first instruction is 1024, (that is, initially `PC=1024` and `nPC=1028`) show the states of `PC` and `nPC` as the four instructions are executed.

- b) If $\%r1 = 4$, $\%r2 = 5$, $\%r3 = 0$, and $\%r4 = 12$ at the start of this code segment, show the states of these four registers as the four instructions are executed.
4. With respect to PC, nPC, the register file and the ALU, describe the steps involved in executing `add %r1, %r2, %r3`. Hint: go through this step by step and don't assume functionality if it isn't on the diagram.