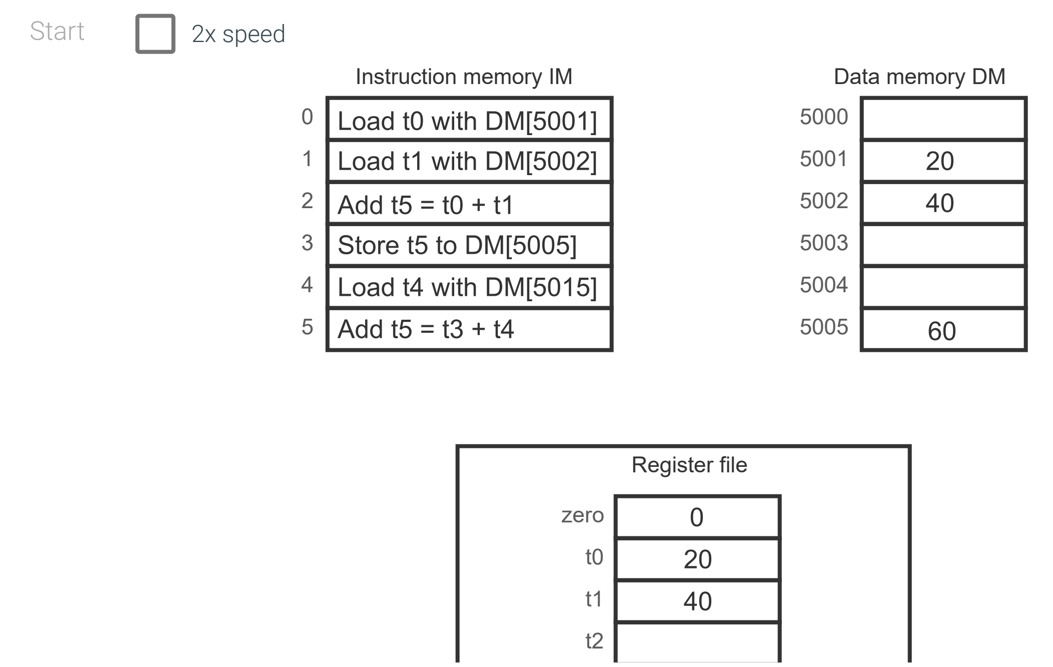
5.1 Programmable processor concept

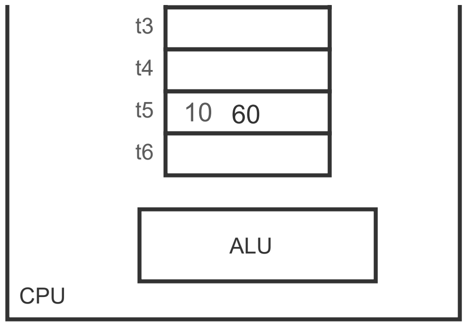
Programmable processor overview

A ***programmable processor*** carries out desired functionality by executing instructions from an instruction memory. Unlike a
that carries out the same functionality repeatedly, a programmable processor can be configured to carry out nearly any fun
putting different instructions in the instruction memory. Ex: A single programmable processor can be programmed to carry
functionality of a calculator, a web browser, and a word processor. Storing instructions into an instruction memory is know
programming the memory, and those instructions are known as a ***program***.

The processor executes one instruction at a time, proceeding to the next instruction when done.

PARTICIPATION ACTIVITY 5.1.1: Simple programmable processor.





A programmable processor includes:

- ***CPU***: A ***central processing unit*** executes instructions by controlling an ALU, register file, and other hardware compone
  - ***ALU***: A component that performs arithmetic and logic operations, like addition or subtraction, on data in the reg for ***arithmetic logic unit***.
  - ***Register file***: A set of registers that holds temporary data accessible by the ALU. A ***register*** is a digital circuit tha multiple bits, such as 32 bits.
- ***Instruction memory***: A memory that holds instructions.
- ***Data memory***: A memory that holds data used by the instructions.

A ***memory*** is a digital circuit that holds relatively large amounts of data, often organized as bytes with each having a unique each byte can either be read ("load") or written ("store"). Ex: A memory may hold 1024 bytes, with addresses 0, 1, 2, ..., 1023 memory sizes range from 1 Kbyte to 4 Gbyte, while typical register files are smaller with perhaps 128 or fewer registers.

PARTICIPATION ACTIVITY 5.1.2: Processor basics.

Refer to the above animation.

1) What is the value in data memory location 5002?

Check Show answer

2) What is the value in t1 after the instruction Load t1 DM[5002] executes?

Check Show answer

3) The first Add instruction adds the values in t0 and t1, and writes the result in what register?

Check Show answer

4) How many executed instructions copied data from the data memory to the register file during the animation?

Check Show answer

5) How many executed instructions used the ALU during the animation?

Check Show answer

6) How many total instructions were

executed during the animation?

Check Show answer

7) The CPU contains a register file and what other component?

Check Show answer

8) A typical register file has 1G or more registers. Type true or false.

Check Show answer

Note that when the processor reads data from the data memory or register file, the data is copied, not removed.

Instructions

A processor executes instructions in sequence, one at a time. The instruction order thus matters.

PARTICIPATION ACTIVITY 5.1.3: Instruction order.

1) Complete the following instruction sequence to add the values in registers t0 and t1 and store the result in DM[5007].

_____

Store t6 to DM[5007]

- ○ Add t2 = t0 + t1
- ○ Add t6 = t0 + t1
- ○ Add t6 = t0 + t2

2) Complete the following instruction sequence to add the values in registers t2 and t6 and store the result in DM[5007].

Add t3 = t2 + t6

_____

- ○ Store t6 to DM[5007]
- ○ Store t3 to DM[5000]
- ○ Store t3 to DM[5007]

3) Select the instruction sequence that calculates the sum of register t1 and DM[5000], and writes the sum in register t5.

- ○ Load t4 with DM[5000]
  Add t5 = t1 + t4
- ○ Add t5 = t1 + t4
  Load t4 with DM[5000]

4) Select the instruction sequence that calculates the sum of t0 and t1, and stores the results to DM[5001].

- ○ Store t2 to DM[5001]
  Add t2 = t0 + t1

○ Add t2 = t0 + t1
Store t2 to DM[5001]

A processor may support hundreds of possible instruction types. Those instruction types can usually be classified into thre

- A ***data transfer instruction*** copies data among the data memory and register file.
- An ***ALU instruction*** operates on data.
- A ***branch instruction*** specifies the location of the next instruction to execute, being different from the next instruction memory.

PARTICIPATION ACTIVITY 5.1.4: Instruction type categories.

Indicate the category for the instruction.

1) Load t0 with DM[5255]

○ Data transfer
○ ALU
○ Branch

2) Jump to instruction 90

○ Data transfer
○ ALU
○ Branch

3) Subtract t6 = t1 - t4

○ Data transfer
○ ALU
○ Branch

Each instruction typically is encoded into a limited number of bits, such as 32 bits. Using a small limited number of bits per ensures more instructions can fit into the memory, and keeps the processor's circuit simple and fast. Some bits may repres instruction type (like Load or Add), other bits may indicate the registers involved (like t0 or t1), and others a data memory a 5005). As such, the number of instruction types is limited. Ex: If the instruction type is represented in 8 bits, then only 2^8 = 2 types are possible.

Thus, a processor's instruction types are limited and kept basic, like the basic Add, Store, and Load instructions seen above must achieve desired functionality using just those relatively-few instruction types.

The set of instruction types supported by a particular processor is called the processor's ***instruction set***. A program written processor's instructions is called an ***assembly language program***, in contrast to programs written in higher-level languages Java, or Python.

PARTICIPATION ACTIVITY 5.1.5: Using limited instruction types.

Assume a processor's only instruction type available for adding is:
Add regA = regB + regC
where regA, regB, and regC each is any register.

1) Which computes t5 = t1 + t3?

- Add t5 = t1 + t3
- Add t1 = t5 + t3

2) Assume the following initial values: t1 = 7, t2 = 6, t3 = 8. What is in t5 after the following:

Add t0 = t1 + t2
Add t5 = t0 + t3

- 13
- 3
- 21

3) Assume the following initial values: t1 = 7, t2 = 6, t3 = 8. What is in t5 after the following:

Add t3 = t1 + t2
Add t5 = t3 + t3

- ○ 26
- ○ 21

4) Which computes t0 = t1 + t2 + t3?

- ○ Add t3 = t1 + t2
  Add t0 = t3 + t3
- ○ Add t4 = t1 + t2
  Add t0 = t4 + t3

5) Which computes t0 = t1 + t2 + t3 + t4?

- ○ Add t4 = t1 + t2
  Add t0 = t4 + t3
- ○ Add t5 = t1 + t2
  Add t6 = t3 + t4
  Add t0 = t5 + t6

6) Can t4 = t3 + t2 + t1 + t0 be computed in two instructions?

- ○ Yes
- ○ No

Register file

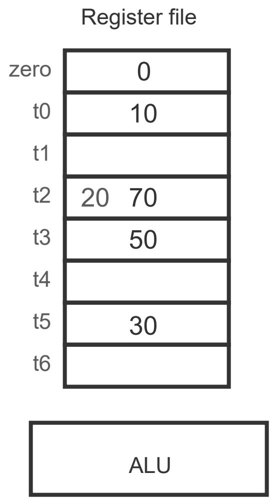
Each register in the register file has a name. The ***zero register*** is a read-only register that always holds the value 0. In the ar
t0 ... t6 refer to the register file's next seven registers, which can be read and written by instructions.

An ALU instruction may read a register's value and write the operation's result into that very same register.

PARTICIPATION ACTIVITY 5.1.6: Register file: Reading and writing.

Start 2x speed

Add t2 = t2 + t3



PARTICIPATION ACTIVITY 5.1.7: Registers.

1) Assume the following initial values: t0 = 7, t1 = 5, and t2 = 3.
What is in t2 after the following:

Add t2 = t2 + t1

- ○ 3
- ○ 8
- ○ 12

2) Assume the following initial values: t0 = 2, t1 = 5, and t2 = 4. What is in t0 after the following:

   Add t0 = t0 + t1
   Add t0 = t0 + t2

   - ○ 2
   - ○ 7
   - ○ 11

3) Assume the following initial values: t2 = 7, t3 = 1, and t4 = 9.
   What is in t4 after the following:

   Add t3 = t3 + t2
   Add t4 = t4 + t3

   - ○ 9
   - ○ 10
   - ○ 17

4) Complete the following instruction sequence to add the values in registers t3, t4, and t5, and store the result in DM[1007].

   Add t6 = t4 + t5

Store t6 to DM[1007]

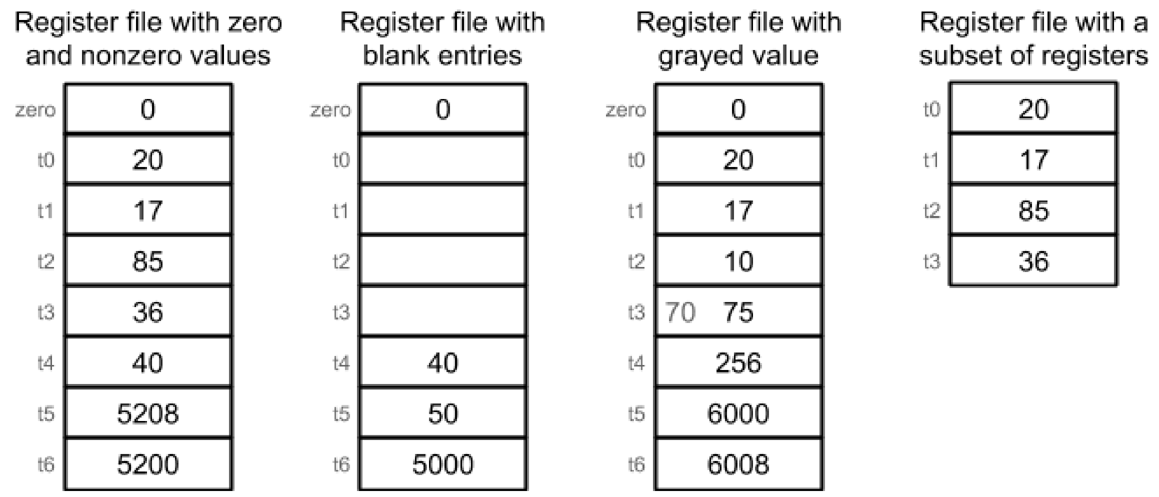
- ○ Add t6 = t6 + t3
- ○ Add t6 = t6 + t4
- ○ Add t6 = t4 + t3

Reset

*A **reset** is an input that when asserted causes a circuit to enter a known state. A processor's reset causes 0's to be written to all registers, including the register file and program counter. So, after the reset, the processor executes the instruction at address 0. A **power-on-reset** circuit resets the processor when power is first applied.*

Register file and data memory

When displaying data values in registers or memory, this material may show: 1) a 0 for register if the register is known to be 0, such as on reset, 2) a blank location if the value is unknown, such as memory location that has not yet been written, or 3) a grayed value representing a previous value for a register or memory location that has been written a new value. Additionally, all registers within the register file may not be shown, instead showing only those registers that are relevant for each example.


Register file with zero and nonzero values
zero 0
t0 20
t1 17
t2 85
t3 36
t4 40
t5 5208
t6 5200
Register file with blank entries
zero 0
t0
t1
t2
t3
t4 40
t5 50
t6 5000
Register file with grayed value
zero 0
t0 20
t1 17
t2 10
t3 70 75
t4 256
t5 6000
t6 6008
Register file with a subset of registers
t0 20
t1 17
t2 85
t3 36

MIPS

MIPS is a processor that was popular in various computers in the 1990's, and is found in some embedded computing devi
is presently one of the most popular processors for learning assembly language programming, and also for learning proce
MIPS is known for having a simple and elegant instruction set, which in turn enables simple and fast processor designs.

MIPS' instruction set has just over 100 instructions, and each instruction is 32 bits. The MIPS register file has 32 registers,
bits. Memory addresses are 32 bits. Memory can be accessed by words (4 bytes), half words (2 bytes), or bytes.

For educational purposes, this material teaches a greatly-simplified version of MIPS, known as ***MIPSzy***, using a small subs
instruction set, and using a register file with only 8 primary registers. MIPSzy only allows memory to be accessed by words

PARTICIPATION ACTIVITY 5.1.8: MIPS and MIPSzy.

1) MIPS is the most popular processor in commercial products today.

○ True

○ False

2) The register file in MIPS has 32

registers.

○ True

○ False

3) The register file in MIPSzy has 32 registers.

○ True

○ False

Exploring further:

- MIPS Processor, Imagination Technologies.
- Computer Organization and Design (5e) - Interactive Version (MIPS)

Provide feedback on this section