General Purpose Microprocessor Design

School of Computer Science and Electronic Engineering
University of Essex (UK)

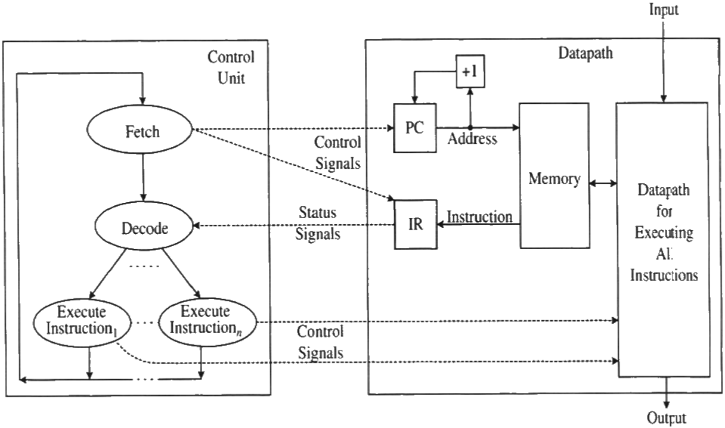
CE339/CE869 - Lecture 8
Jan 2020

Outline

- General Purpose Microprocessor
- General Datapath
- Control Unit
- Example

Overview of the CPU Design

- The CPU is simply a dedicated microprocessor that only executes software instructions.



CPU Design

- Instruction set
  - How many instructions (CISC vs RISC)?
  - What are the instructions?
  - Fixed (e.g. ARM) vs variable (e.g. x86) length instruction set?
  - What is opcode for each instruction?
- Datapath
  - What functional units do you want?
  - General purpose register file or accumulator?
  - How are the different units connected together?
  - In a general purpose microprocessor, there is a program counter (PC) and an instruction register (IR).

Types of instructions [1]

- Arithmetic and logic operations
  - Add, subtract, multiply, or divide the values of two registers, placing the result in a register, possibly setting flags in status register.
  - Perform bitwise operations, e.g., AND, OR, XOR of corresponding bits in a pair of registers; taking the negation of each bit in a register.
  - Compare two values in registers ($>$, $<$, $=$, $\leq$, $\geq$).

[1] From Wikipedia: `http://en.wikipedia.org/wiki/Instruction_set`

Types of instructions [1]

- Arithmetic and logic operations
  - Add, subtract, multiply, or divide the values of two registers, placing the result in a register, possibly setting flags in status register.
  - Perform bitwise operations, e.g., AND, OR, XOR of corresponding bits in a pair of registers; taking the negation of each bit in a register.
  - Compare two values in registers ($>$, $<$, $=$, $\leq$, $\geq$).
- Data handling and memory operations
  - Set a register to a fixed constant value.
  - Move data from a memory location to a register, or vice versa. Used to store the contents of a register, result of a computation, or to retrieve stored data to perform a computation on it later.
  - Read and write data from hardware devices.

[1] From Wikipedia: `http://en.wikipedia.org/wiki/Instruction_set`

Types of instructions [1]

- Arithmetic and logic operations
  - Add, subtract, multiply, or divide the values of two registers, placing the result in a register, possibly setting flags in status register.
  - Perform bitwise operations, e.g., AND, OR, XOR of corresponding bits in a pair of registers; taking the negation of each bit in a register.
  - Compare two values in registers ($>$, $<$, $=$, $\leq$, $\geq$).
- Data handling and memory operations
  - Set a register to a fixed constant value.
  - Move data from a memory location to a register, or vice versa. Used to store the contents of a register, result of a computation, or to retrieve stored data to perform a computation on it later.
  - Read and write data from hardware devices.
- Control flow operations
  - Branch (jump) to another location in the program.
  - Conditionally branch to another location if a certain condition holds.
  - Indirectly branch to another location, while saving the location of the next instruction as a point to return to (a "call to a function").

[1] From Wikipedia: `http://en.wikipedia.org/wiki/Instruction_set`

Outline

- General Purpose Microprocessor
- General Datapath
- Control Unit
- Example

General Datapath

- Its role is performing the data operations for all the instructions.
- Its design is an incremental process: a class of instructions is considered at each increment and a portion of the datapath is built.
- The complete datapath is generated by combining partial datapaths together.
- Datapath design: how control signals affect flow of data and function of data units (not how control signals are generated, which is "control unit" design).

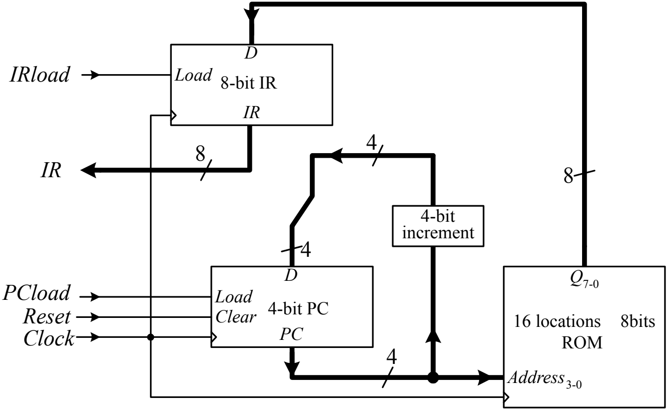
Incremental Design Process

- **Step 1: program sequencing - instruction fetching**. It involves an instruction register(IR), a program counter (PC), and an adder for incrementing the PC.
- **Step 2: "arithmetic/logic instruction" datapath**. It involves an ALU and register file / accumulator plus other components such as muxes and shifters.
- **Step 3: "data transfer instruction" datapath**. It involves a data memory and the register file / accumulator. The location of data is from the address field of the instruction.
- **Step 4: "control flow instruction" datapath**. A path from the address field of the instruction to the PC input.

Step 1: program sequencing

Implementation of very simple "program sequencing datapath"

(assuming all instructions are 8 bits wide and that the program fits in a 16 bytes ROM)



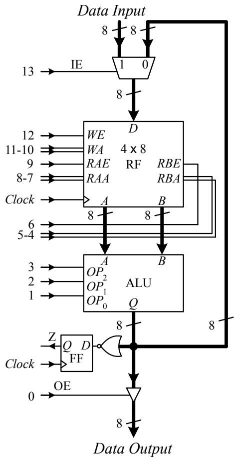
At the clock:

IRload asserted ⇒ fetches next instruction and stores it in Instruction Register

PCload asserted ⇒ Program Counter increments (points to next instruction)

Step 2: "arithmetic/logic instruction" datapath

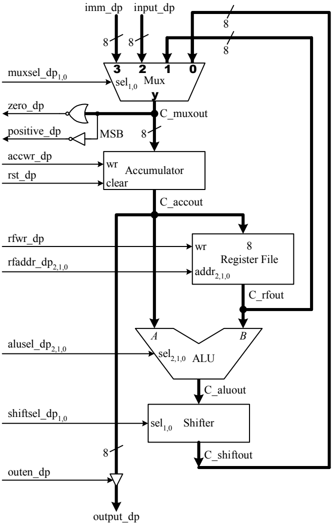
General purpose register file or accumulator?



Gen. purpose register file (ARM)
more versatile:
ADD Rd, Ra, Rb;
Rd ← Ra + Rb
but longer opcodes
(they need to specify 3 registers)

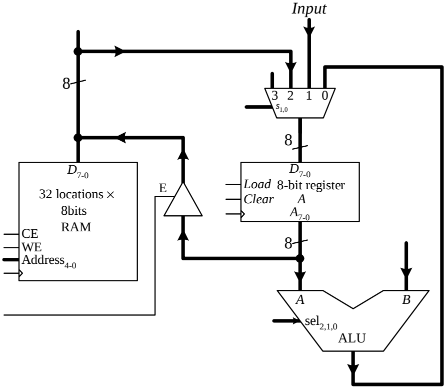
With accumulator (PIC)
less versatile:
ADD A, Rb;
A ← A + Rb
which involves more MOV/LOAD-/STORE instructions
but shorter opcodes
(they need to specify 1 register)

2-operand gen. purp. r.f. (x86)
compromise:
ADD Rd, Rb;
Rd ← Rd + Rb
no accumulator but dest. and left operand coincide



Step 3: "data transfer instruction" datapath

Needs to implement a way to exchange data between the registers and the main memory. Example of simple implementation:

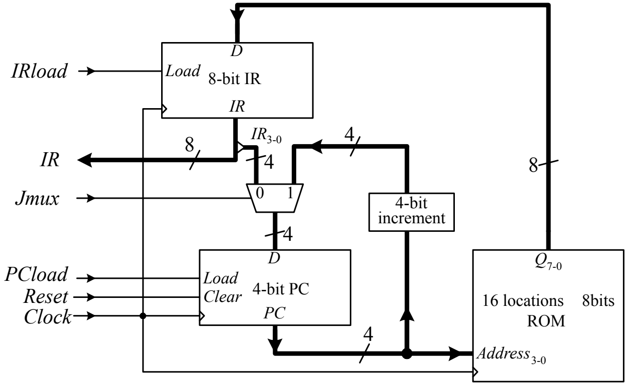


In this case the accumulator can receive input from the result of the operation, from memory or from *input*.

The value in the accumulator can be saved in RAM.

Step 4: "control flow instruction" datapath

Needs to implement a way to load the *PC* with the JUMP or BRANCH address. Example of simple implementation:



the *Jmux* signal selects whether to increment *PC* or load the 4 LSB from the opcode (IR_{3-0})

Outline

- General Purpose Microprocessor
- General Datapath
- Control Unit
- Example

Control Unit

- Cycles through three main steps (instruction cycle):
  - **Step 1**: fetches an instruction - reads the memory location specified by the PC and copies the content of that location into the IR.
  - **Step 2**: decodes the instruction - extracts the opcode bits from the IR and determines what the current instruction is.
  - **Step 3**: executes the instruction - asserts the appropriate control signals.

Hardwired VS Microcode Control Unit

- Hardwired Control Unit:
  - Implemented as a FSM that cycles through the fetch/decode/execute steps (instruction cycle).
  - Advantages: simpler for very simple microprocessors; can operate at high speed.
  - Disadvantages: little flexibility; difficult to design and debug; does not scale to more complex (modern) microprocessors.

Hardwired VS Microcode Control Unit

- Hardwired Control Unit:
  - Implemented as a FSM that cycles through the fetch/decode/execute steps (instruction cycle).
  - Advantages: simpler for very simple microprocessors; can operate at high speed.
  - Disadvantages: little flexibility; difficult to design and debug; does not scale to more complex (modern) microprocessors.
- Microcode Control Unit:
  - The processor's behaviour and programming model defined via microprograms, i.e. sequences of microinstructions that drive the control signals to execute instruction cycles, e.g. Fetch, Indirect, Execute, and Interrupt.
  - Advantages: can scale to more complex microprocessors; easier to fix bugs; implement specialized microprocessors (different instruction sets) with same underlying hardware micro-architecture.

Outline

- General Purpose Microprocessor
- General Datapath
- Control Unit
- Example

Instruction Set

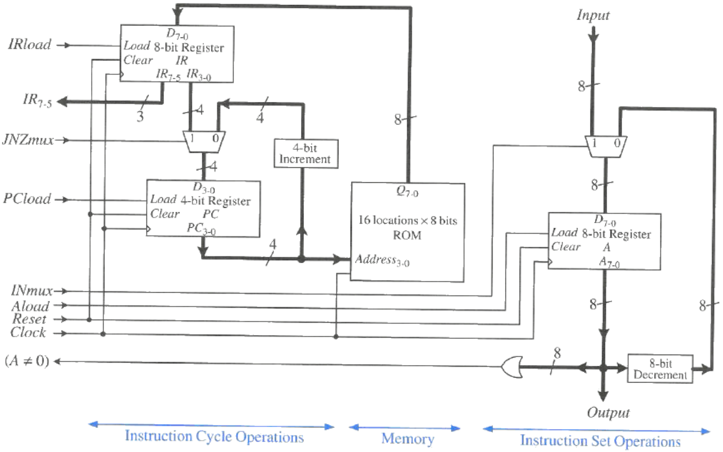
Extremely simple example: the ALU is reduced to a single operation, decrement (not too useful, just a toy example)

- Five instructions, three bits are required to encode the operation field of the opcode.

Instruction	Encoding	Operation
IN A	011xxxxx	A$\leftarrow$ Input
OUT A	100xxxxx	Output$\leftarrow$ A
DEC A	101xxxxx	A$\leftarrow$ A-1
JNZ address	110xaaaa	IF (A!=0) THEN PC=aaaa
HALT	111xxxxx	Halt

- aaaa = four bits for specifying a memory address (assuming 16 bytes memory)
- x = don't care
- A = accumulator

The Datapath

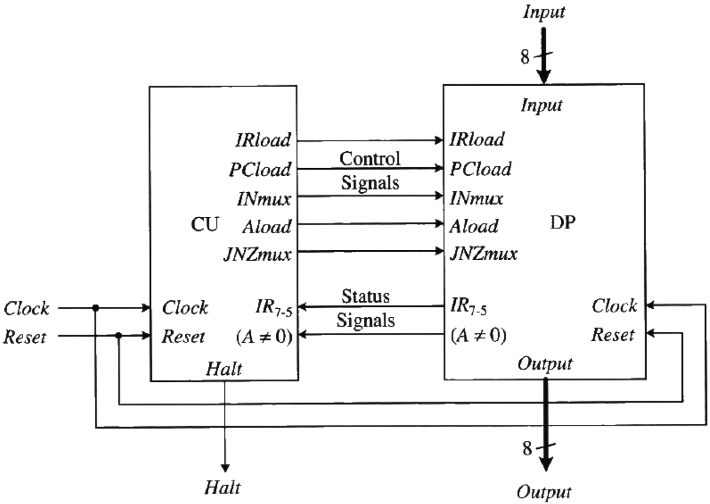


The Datapath

- Program stored in 16x8 ROM, the 4-bit address is required. The PC is 4-bit wide.
- Each instruction is 8-bit wide, the IR is 8-bit wide.
- A 4-bit increment unit is used for the PC.
- The PC needs to be loaded with either the result of the increment unit or the address from JNZ instruction.
- The output of the PC is connected directly to the 4-bit memory lines.
- The 8-bit memory output is connected to the input of the IR.
- The JNZ instruction requires an 8-bit OR gate connected to the output of the accumulator to test the condition ($A \neq 0$).
- Five control signals: IRload, PCload, INmux, Aload, and JNZmux.
- One status signal: ($A \neq 0$)

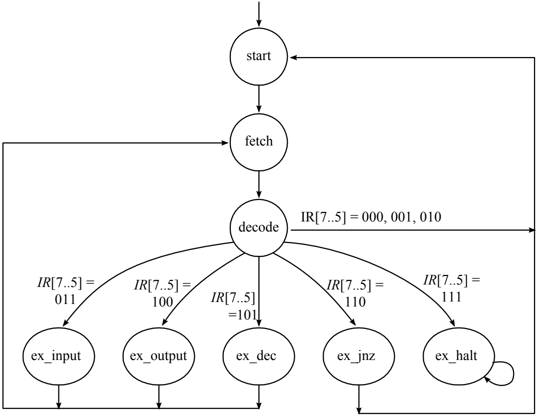
The Complete Circuit

- The datapath and control unit.



The Control Unit

- Its design starts from a state diagram:



The Control Unit

- The first state, "start", serves as the initial reset state. No action is performed.
- It requires an extra clock cycle.
- JNZ instruction requires an extra clock cycle to complete its operation, because the PC must be loaded with a new address value if the condition is tested true. This new address value is loaded into the PC at the next clock cycle.

The Control Unit

- The next step is the next state table from the state diagram.

curr. state	$IR_7 IR_6 IR_5$				
	011 INPUT	100 OUTPUT	101 DEC	110 JNZ	111 HALT
start	fetch	fetch	fetch	fetch	fetch
fetch	decode	decode	decode	decode	decode
decode	ex_input	ex_output	ex_dec	ex_jnz	ex_halt
ex_input	fetch	fetch	fetch	fetch	fetch
ex_output	fetch	fetch	fetch	fetch	fetch
ex_dec	fetch	fetch	fetch	fetch	fetch
ex_jnz	start	start	start	start	start
ex_halt	ex_halt	ex_halt	ex_halt	ex_halt	ex_halt

The Control Unit

- And the output table from the state diagram:

Control word	State	IRload	PCload	INmux	Aload	JNZmux	Halt
0	start	0	0	0	0	0	0
1	fetch	1	1	0	0	0	0
2	decode	0	0	0	0	0	0
3	ex_input	0	0	1	1	0	0
4	ex_output	0	0	0	0	0	0
5	ex_dec	0	0	0	1	0	0
6	ex_jnz	0	1 or 0	0	0	1	0
7	ex_halt	0	0	0	0	0	1

A Sample Program

- A loop sample program is:

```
      IN  A
loop: OUT A
      DEC A
      JNZ loop
      HALT
```

```
Memory    Instruction
address   encoding
0000      01100000 -- IN A
0001      10000000 -- OUT A
0010      10100000 -- DEC A
0011      11000001 -- JNZ
0100      11111111 -- HALT
```

Thank you
for your attention

Questions?

General Purpose Microprocessor Design

School of Computer Science and Electronic Engineering
University of Essex (UK)

CE339/CE869 - Lecture 8
Jan 2020