1 Motivation

2 Basic Concepts

Microinstructions

Microprogrammed Control Unit

Advantages and Disadvantages

$\mu$-Instruction sequencing

Two address fields

Two address fields

Variable Format

$\mu$-Instruction Execution
Table of Contents I

3 Where to focus your study

4 Where to focus your study
Remember what we talked about in the last class?
Motivation

Remember what we talked about in the last class?

- **µ-operations** - the most basic instruction executed by a processor:
- **Control signals** - to send data from the various locations;
- **Control unit operation** - how to control signals and micro-operations.
We saw that it was possible to operate the control unit through:

- An hardwired version:
  - Complete boolean circuit;
  - Difficult to implement for complex systems.
  - Impossible to add an instruction after implementing the circuit.

So what is the alternative? Any ideas?
Alternative: $\mu$-programmed control unit

- The logic of the control unit is specified by a microprogram;
  - Also known as firmware.
- A microprogram consists of a sequence of $\mu$-operations.
How can we use the concept of microprogramming to implement a control unit?

For each \( \mu \)-operation:

- Control unit is allowed to generate a set of control signals;
- Each control line is either on / off;
- This state can be represented by a binary digit for each control line.
**Idea:** construct a control word (a.k.a. $\mu$-instruction)

- Bit represents one control line
- Each $\mu$-op would be represented by a different pattern of 1s and 0s.
Format of the $\mu$-instruction or control word is as follows:

- One bit for each internal processor control line;
- One bit for each system bus control line;
- Condition field indicating the condition under which there should be a branch (JMP);
- Field with the address of the $\mu$-instruction to be executed next when a branch is taken.
Such a microinstruction is interpreted as follows:

- Activate control lines indicated by a 1 bit;
- Deactivate control lines indicated by a 0 bit;
- If condition indicated by the condition bits is
  - **False:** execute the next microinstruction in sequence;
  - **True:** execute the instruction indicated in the address field.
How can we process these $\mu$-instructions?

Processing a $\mu$-program requires:

- Going through a sequence of $\mu$-instructions;
- Jumping to other addresses of $\mu$-instructions;
- I.e. normal procedure for processing instructions:
  - Only difference is that these instructions are comprised of $\mu$-operations.
Let's take a look at one possible architecture:

![Control Unit Microarchitecture Diagram](Source: [Stallings, 2015])
• **Control memory:**
  - Containing the set of $\mu$-instructions;

• **Control address register (CAR):**
  - Containing the address of the next $\mu$-instruction to be read;
  - Remember MAR?

• **Control buffer register (CBR):**
  - Containing the $\mu$-instruction read from the control memory;
  - After $\mu$-instruction is read it can be executed;
  - Remember MBR?

• **Sequencing logic:**
  - Loads CAR and issues a read command.
In greater detail:

Figure: Function of microprogrammed control unit (Source: Stallings, 2015)
The control unit functions as follows to execute an instruction:

1. **Sequencing logic unit:**
   - Loads an address to be read into CAR;
   - Issues a READ command to the control memory;

2. **Word is read into CBR;**

3. **Content of the CBR generates:**
   - Control signals for the CPU;
   - Next address for the sequencing logic unit;

4. **Sequencing logic unit:**
   - Loads a new address into the CAR;
   - Based on the next-address information from the CBR and the ALU flags.

All this happens during one clock pulse.
Does all of this sound strangely familiar? Sense of *déjà vu*?
Does all of this sound strangely familiar? Sense of déjà vu?

- Similar to how control unit processed different stages of an instruction.
- Remember?
  - MAR
  - MBR
  - PC
  - Fetch cycle
  - Indirect cycle
  - Interrupt cycle
- But now are doing it in a "smaller" scale within the control unit:
  - Instead of a system-wide: processor, bus, ram...
Advantages and Disadvantages

Main **advantage** of the use of $\mu$-programming:

- Simplifies the design of the control unit;
  - Both cheaper and less error prone to implement.
- Hardwired control unit must contain complex logic:
  - Microprogrammed control unit components are simple pieces of logic.

Main **disadvantage** of a $\mu$-programmed unit:

- Slower than a hardwired unit of comparable technology;
Tasks performed by a microprogrammed control unit are:

- **μ-instruction sequencing:**
  - Get next μ-instruction from the control memory.

- **μ-instruction execution:**
  - Generate control signals needed to execute the μ-instruction.

Let's focus on the first one.
Address of next \( \mu \)-instruction to be executed is:

- Determined by instruction register or;
- Next sequential address or;
- Branch (JMP).

Decision is based on:

- Current \( \mu \)-instruction;
- Condition flags;
- Contents of the instruction register;
Wide variety of techniques to generate the next $\mu$-instruction address:

- Two address fields
- Single address field
- Variable format
Two address fields

Figure: Branch Control Logic: Two Address Fields (Source: Stallings, 2015)
Simplest approach: provide two address fields:

- Multiplex between both address fields and IR:
  - Updating CAR accordingly.
- CAR is decoded to produce the next $\mu$-instruction address;
- Branch logic module selects the address-selection signals:
  - Based on control unit flags and
  - Bits from the control portion of the $\mu$-instruction.
Single address field
Two-address approach is simple but:

- Requires more bits in the \( \mu \)-instruction than other approaches.

**Idea:** Use additional logic to have only one address field;

Options for next address are as follows:

- Address field;
- IR code;
- Next sequential address (+1).
However:

Often, the address field will not be used... Can we do better? Any ideas?
Variable Format

Figure: Branch Control Logic: Variable Format (Source: Stallings, 2015)

Figure 1: Branch Control Logic: Variable Format.
Provide for two entirely different microinstruction formats:

- **Format 1**: bits are used to activate control signals:
  - Next address is either: \{next sequential address, address derived from IR\}.

- **Format 2**: some bits drive the branch logic, remaining provide address:
  - Either a conditional or unconditional branch is being specified.
Tasks performed by a microprogrammed control unit are:

- **µ-instruction sequencing:**
  - Get next microinstruction from the control memory.

- **µ-instruction execution:**
  - Generate control signals needed to execute the µ-instruction.

Let's focus on the second one.
From the previous classes:

Do you remember what is the function of the control unit?
Basic Concepts

µ-Instruction Execution

Remember this?

![Data paths and control signals](Source: [Stallings, 2015])

Execution of a µ-instruction: generate control signals.

- Some signals control points internal to the processor;
- Other signals go to the system bus;
We can now update our previous Figure 3:

Figure: Function of microprogrammed control unit (Source: [Stallings, 2015])
With this one:

Figure: Control Unit Organization (Source: (Stallings, 2015))
What is the difference between pictures?

First picture focused on the sequencing logic module:

- Containing logic to generate address of next $\mu$-instruction using:
  - as inputs the IR, ALU flags, CAR, CBR.
- Driven by a clock that determines the timing of the $\mu$-instruction cycle.

Second picture introduces control logic module:

- Generates control signals as a function of the $\mu$-instruction;
Control signals can be transmitted in various ways (1/2):

- $K$ control signals:
  - Can be controlled using $K$ lines, allowing for $2^k$ possibilities;
  - Not all of these possibilities are valid, e.g.:
    - Two sources cannot be gated to the same destination;
    - A register cannot be both source and destination;
    - Only one pattern of control signals can be presented to the ALU at a time.
    - Only one pattern of control signals can be presented to the external control bus at a time.
  - We can do better than this...
Control signals can be transmitted in various ways (2/2):

- Let $Q$ represent all allowable combinations of control signals:
  - Possible combinations: $Q$ with $Q < 2^K$ possibilities;
  - We can encode these $Q$ combinations using $\log_2 Q$ bits;
  - Therefore $\log_2 Q < 2^K$
Advantages / Disadvantages:

- **Unencoded format:**
  - Advantage:
    - Little or no decode logic is needed;
    - Each bit generates a particular control signal.
  - Disadvantage:
    - Requires more bits than necessary.

- **Encoded format:**
  - Advantage:
    - Requires less bits.
  - Disadvantage:
    - Requires complex logic to encode / decode resulting in loss of performance.
Example (1/5)

Assume a processor with:

- Single accumulator register;
- Several internal registers:
  - Such as a program counter and a temporary register for ALU input.
- Instruction format where:
  - First 3 bits indicate the type of operation;
  - Next 3 encode the operation;
  - Final 2 select an internal register
Simple register transfers

Figure: Simple register transfers (Source: Stallings, 2015)
Example (3/5)

Figure: Memory Operations (Source: (Stallings, 2015))
Example (4/5)

* CSAR = channel system address register, special register for controlling bus lines that exists in some processors.

Figure: Special sequencing operations (Source: (Stallings, 2015))
Example (5/5)

ALU operations

- 0 1 1 0 0 0
  - ACC ← ACC + Register

- 0 1 1 0 0 1
  - ACC ← ACC - Register

- 0 1 1 0 1 0
  - ACC ← Register

- 0 1 1 0 1 1
  - Register ← ACC

- 0 1 1 1 0 0
  - ACC ← Register + 1

Register select

Figure: ALU operations (Source: Stallings, 2015)
After this class you should be able to understand that:

- Execution of an instruction involves the execution of substeps:
  - Each cycle is in turn made up of $\mu$-operations;
- Control unit causes the processor to go through a series of $\mu$-operations:
  - in the proper sequence;
  - and generating the appropriate control signals;
After this class you should be able to understand that:

- Alternative to a hardwired control unit is a $\mu$-programmed control unit:
  - logic is specified by a $\mu$-program:
    - which consists of a sequence $\mu$-operations.
  - $\mu$-programmed control unit is a simple logic circuit capable of:
    - sequencing through $\mu$-instructions;
    - generating control signals to execute each $\mu$-instruction.

- As in a hardwired control unit:
  - Control signals generated by a $\mu$-instruction are used to cause register transfers and ALU operations.
Less important to know how these solutions were implemented:

- details of specific hardware solutions.

Your focus should always be on the building blocks for developing a solution =)

*Computer Organization and Architecture.*

Pearson Education.