TYPICAL MICROPROGRAMMED CONTROL UNIT *(Super Important!)*

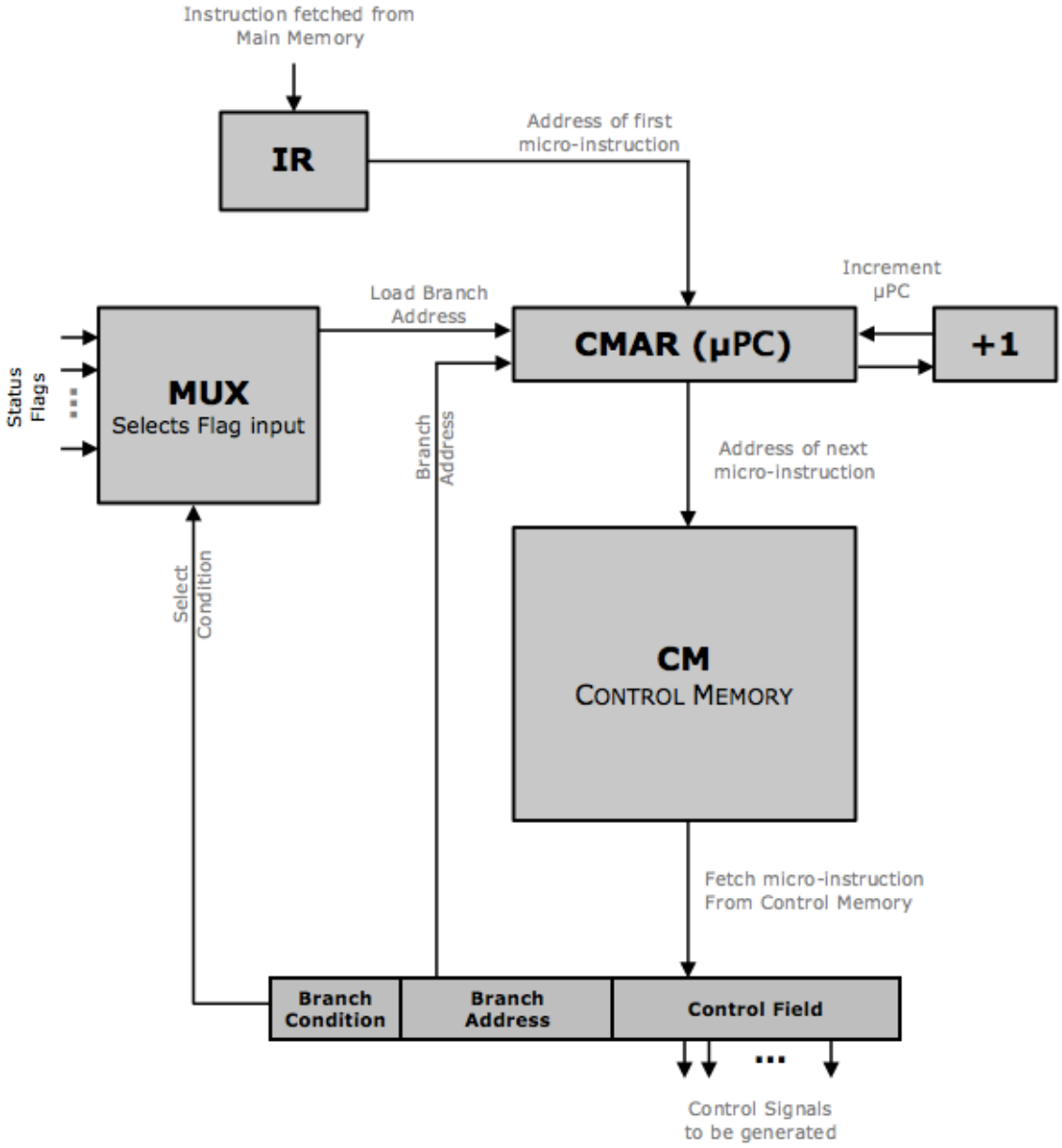
1) Microprogrammed Control Unit produces control signals by **software**, using **micro-instructions**.
2) A program is a set of instructions.
3) An instruction requires a set of Micro-Operations.
4) **Micro-Operations are performed by control signals**.
5) Here, these control signals are generated using **micro-instructions**.
6) This means **every instruction requires a set of micro-instructions**
7) **This is called its micro-program**.
8) Microprograms for all instructions are **stored in a small memory called "Control Memory"**.
9) The Control memory is **present inside the processor**.
10) Consider an **Instruction** that is **fetched from the main memory** into the Instruction Register (**IR**).
11) The processor uses its unique **"opcode"** to identify the **address of the first micro-instruction**.
12) That address is loaded into **CMAR** (Control Memory Address Register) also called **μIR**.
13) This address is **decoded** to **identify the corresponding μ-instruction** from the Control Memory.
14) There is a big **improvement over Wilkes' design**, to reduce the size of micro-instructions.
15) Most micro-instructions will **only have a Control field**.
16) The **Control field** Indicates the **control signals to be generated**.
17) Most micro-instructions **will not have an address field**.
18) Instead, **μPC will simply get incremented** after every micro-instruction.
19) This is as long as the μ-program is executed **sequentially**.
20) If there is a **branch μ-instruction** only then there will be an **address filed**.
21) If the branch is **unconditional**, the **branch address will be directly loaded into CMAR.**
22) For **Conditional branches**, the Branch condition will **check the appropriate flag**.
23) This is done using a **MUX** which has all flag inputs.
24) If the **condition is true**, then the **MUX will inform CMAR to load the branch address**.
25) If the **condition is false** CMAR will simply get **incremented**.
26) The control memory is usually implemented using **FLASH ROM** as it is **writable** yet **non volatile**.

ADVANTAGES

1) The biggest advantage is **flexibility.**
2) Any **change** in the control unit can be performed by **simply changing the micro-instruction**.
3) This makes **modifications and up gradation** of the Control Unit **very easy**.
4) Moreover, software can be **much easily debugged** as compared to a large Hardwired Control Unit.
5) Since most micro-instructions are executed sequentially, they don't **need for an address field**.
6) This significantly **reduces the size of micro-instructions**, and hence the **Control Memory**.

DRAWBACKS

1) **Control memory** has to be present **inside** the processor, **increasing its size.**
2) This also **increases the cost** of the processor.


Instruction fetched from Main Memory
IR
Address of first micro-instruction
Increment µPC
Load Branch Address
CMAR (µPC)
+1
Status Flags
MUX
Selects Flag input
Branch Address
Address of next micro-instruction
Select Condition
CM
CONTROL MEMORY
Fetch micro-instruction From Control Memory
Branch Condition
Branch Address
Control Field
Control Signals to be generated

APPLICATIONS OF MICROPROGRAMMED CONTROL UNIT / MICROPROGRAMMING

Microprogramming has various advantages like flexibility, simplicity, cost effectiveness etc.
As a result, it plays a major role in the following applications.

1) DEVELOPMENT OF CONTROL UNITS

Modern processors have very **large and complex instruction sets**.
Microprogramming is ideally suited for **making Control Units** of such processors as they are far less complex and can be **easily modified**.

2) USER TAILORING OF THE CONTROL UNIT

As the Control Unit is developed using software, it can be **easily reprogrammed**.
This can be used for **custom made modifications** of the Control Unit.
To allow this, the Control Memory must be **writeable like RAM or Flash ROMs**.

3) EMULATION

Emulation is when one **processor (A) is made to emulate or behave like another processor (B)**.
To do this, "A" must be able to execute the instructions of "B".
If we re-program the Control Memory of "A", same as that of "B", then "A" will be able to emulate the behavior of "B", for every instruction. This is possible only in Microprogrammed Control Units.
Used generally when a **main processor** has to **emulate** the behavior of a **math co-processor**.

4) IMPROVING THE OPERATING SYSTEM

Microprogramming can be used to implement **complex and secure functions of the OS**.
This not only makes the OS **more powerful and efficient**, but more importantly **secure**, as it provides the OS a higher degree of protection from malicious **virus attacks**.

5) HIGH LEVEL LANGUAGE SUPPORT

Modern high level languages have **far more advanced and complex data types**.
Microprogramming can **provide support for such data types** directly from the processor level.
As a result the language becomes **easy to compile and also faster to execute**.

6) MICRO-DIAGNOSTICS

As Microprogrammed Control Units are software based, **debugging an error is far more easy** as compared to doing the same for a complex hardwired control unit. This allows **monitoring, detection, isolation and repairs** of any kind of **system errors** in the Control Unit.
At times, it can also act as a **runtime substitute**, if the corresponding **hardwired component fails**.

7) DEVELOPMENT OF SPECIALIZED PROCESSORS

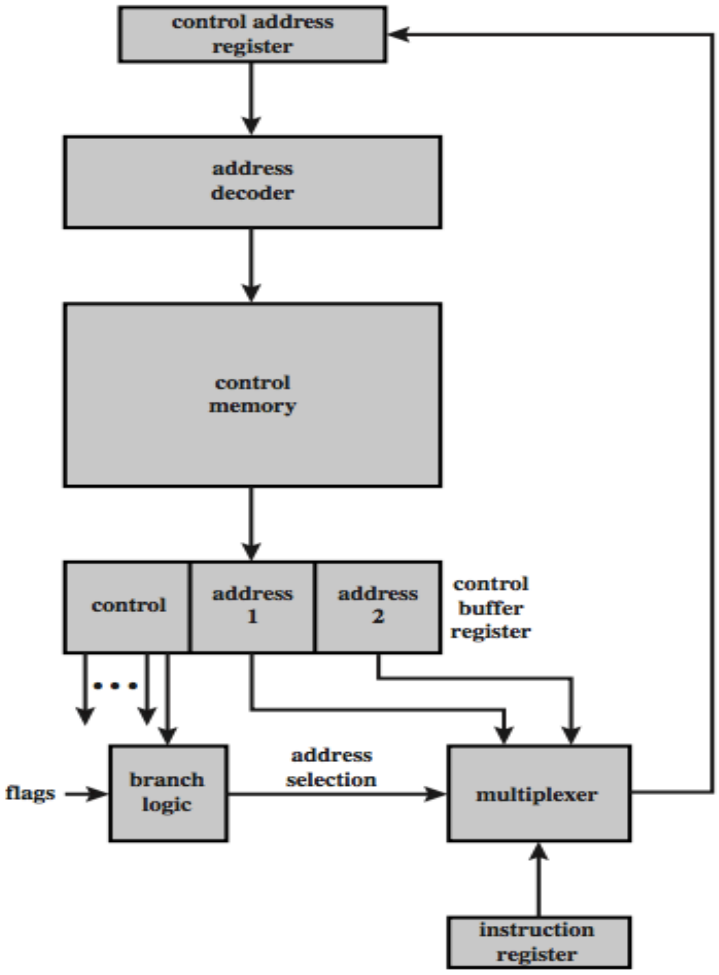
All processors are not general purpose. Many applications require **specialized processors** like **DSP** (Digital Signal Processors) for **communication**, **GPU** (Graphic Processor Unit) for **image processing**.
They have **complex instruction sets** and also need to be **constantly upgraded**, making Microprogrammed Control Unit the **ideal choice**.

MICRO-INSTRUCTION SEQUENCING

It is the method of determining the flow of the Microprogram.
There are two main techniques.

1) DUAL ADDRESS FIELD

In this approach, micro-instructions are **NOT executed in a sequential manner**.
The instruction register (**IR**) gives the address of the **first micro-instruction**.
Thereafter, **each micro-instruction gives the address of the next micro-instruction**.
If it is a **conditional micro-instruction**, it will contain **two address fields**.
One for the condition to be **true** and the other for **false**.
Hence it is called **dual address field**.
The **multiplexer will decide the address** the address to be loaded into the control memory address register (CMAR) **based on the status flags**.



2) SINGLE ADDRESS FIELD

In this approach, micro-instructions **are executed in a sequential manner**.
The instruction register (**IR**) gives the address of the **first micro-instruction** into CMAR.
Thereafter the **address is simply incremented**.
Hence every micro-instruction **need not carry address** of the next one.
This is true so long as the μ-program is executed in a **sequential manner**.
For an **unconditional branch**, the micro-instruction **contains the branch address**.
This address will be **loaded into CMAR**.
For a **conditional branch**, the micro-instruction contains the **branch address for true condition**.
If the **condition is false**, the current address in **CMAR will be simply incremented**.
This means even in the **worst case**, the micro-instruction will **carry only one address**.
Hence it is called **single address field**.
The **multiplexer will decide the address** the address to be loaded into the control memory address register (CMAR) **based on the status flags**.