Nixdorf Computers

History and Hardware

The predecessor to the 8870 Family was the 820/20 Family.
The 820/20 used a 3 or 4 slot Processor module (LFI 15x), a Memory module (LFI 16x), a Micro Program module (LFI 17x), one or two Macro Program modules (LFI 17x), and depending on the connected peripherals, maybe an I/O module (LFI 18X). All modules, except for the Processor were one slot modules.
There were two Buses on the Backplane of the chassis, an Address Bus and a Data Bus to which all modules were connected. There were also a number of control signals used – some common to all modules, and some slot specific. Only one module could have control of the Backplane at any one time.
The computer was an asynchronous computer. There was no Master Clock.
Each module had its own Timing Chain.

The processor would apply an Address onto the Address Bus, generate the required Start signal, and then wait. The particular module would pick up the Start Signal which would then start the module's internal Timing Chain. The module would hold a control signal low to signify that it was working, and would release the control signal once the module had finished its activity. The release of the control signal would trigger the Processor to restart its activity.
If Data was to be conveyed between the Processor and a module, the Data would be placed on the Data Bus, and at the appropriate time, a Strobe control signal would be applied to the Backplane, once the Data signals had stabilised.

The Micro and Macro modules were fairly simple modules, as they were ROM modules. Once the Start signal was applied to the module, the information on the Address Bus was loaded into the module, and the selected line was then read. The information from that line would be placed on the Data Bus, and a Strobe control signal issued. Shortly afterwards, the module would release the control signal that it had held low whilst its Timing Chain had functioned.

The Memory modules used Destructive Read, thus having two parts during its operation – a Read function, followed by a Write function. Unless new information was to be stared in cell, the Data read from a cell would be stored back into the cell.

The I/O modules were divided into two parts, an Input Part and an Output Part with the Processor being able to specify whether an Input or an Output function was required. A fairly robust cable was used to connect the I/O module to the peripherals.
The Processor had an I/O module (one of its slots), and the Cable was used to connect to the 10 Key Operator Keyboard and the Printing device.

The Processor

In the Processor were a small number of 12 bit Registers.
The A Register.
The Micro Program Address Register.
The Address Part Of the current Micro Program instruction.

There was also a 6 bit Register, used to hold the Operation Part of the current Micro Program instruction.

There was also a 1 bit Register referred to as the Carry Bit. This was used as the overflow (or a 13th bit) of the A Register.

The Micro Program

Each Micro Program instruction was a fixed length of 18 bits.
6 bits for the Operation Part, and the remaining 12 bits for the Address Part.

The Macro Program

Each Macro Program instruction was also a fixed length or 18 bits.
6 bits for the Operation Part, 1 bit used for Indexing, and the remaining 11 bits for the Address Part.
Examples of Macro Instructions =

- Print an "A" at the current location on the Printer.
- Turn the on Red Light as the Operator had made an input error.
- Position the Print Head of the Printer to location 15.

The Operation Part for a Print a Character command was a 2.F hexadecimal.
The Operation Part for a Turn on the Error Lamp command was a 2.C hexadecimal.
The Operation Part for a Position the Print Head command was a 2.D hexadecimal.

As will be seen, the Operation Part of 6 bits can range from 0.0 to 3.F (hexadecimal), although not every number was used.
This is all the information that I have about the Macro Instructions.

Basically the concept of the Nixdorf Computer was to have a number of fixed routines written in the Micro Program, with one routine for each Macro Program instruction. The routine for a Macro Instruction was accessed via an entry in a 64 location Table in the Micro.
The Micro Program would also have a number of internal Routines, such as what to do if there was a Power Failure, picking up the numbers from the Keyboard and placing them in the Input Buffer, checking the status of the 5 millisecond clock (used when magnets need to be energised), etc.

The Micro Program Instruction Set

As previously stated, a Micro Program instruction was made up of 18 bits, 6 for the OP and 12 for the Addr.

OP Code

The most significant 2 bits of an OP Code (bits 5 and 6) are modifiers.
The least significant 4 bits of an OP Code (bits 1 to 4) are the type of Operation that the Processor will perform.

OP Code bits 5 and 6 = 0.0 (00).

OP Code bits =

Value	What it did
0	Used for Incremental(+1) activity.
1	Used for Subtraction (-1) activity.
2	Used for Storage or Macro Instruction Fetching.
3	Used to Left Shift the contents of the A Register 1 time (effectively doubling the value in the A Register, with bit 12 going to the **Carry Bit**, and the **Carry Bit** becoming bit 1 of the A Register.
4	Load the value in the Addr Part (bits 1 to 12 of the Instruction) into the A Register.
5	Cyclic Sum the value in the Addr Part (bits 1 to 12 of the Instruction) with the contents of the A Register, and store the result back into the A Register.
6	Logical And the value in the Addr Part (bits 1 to 12 of the Instruction) with the contents of the A Register, and store the result back into the A Register.
7	Reset the **Carry Bit**, Add the value in the Addr Part (bits 1 to 12 of the Instruction) to the contents of the A Register, and store the result back into the A Register, and set the **Carry Bit** if necessary.
8	Jump to the Address specified in the Addr Part (bits 1 to 12 of the Instruction) **IF** the contents of the A Register <>0.
9	Jump to the Address specified in the Addr Part (bits 1 to 12 of the Instruction) **IF** the contents of the A Register = 0.
A	Jump to the Address specified in the Addr Part (bits 1 to 12 of the Instruction) **IF** the contents of the A Register bits 1 to 4 are greater than a decimal digit.
B	Jump to the Sub-Routine commencing at the Address specified in the Addr Part (bits 1 to 12) of the Instruction, and place the Address of this JSR instruction + 1 into the A Register.
C	Cannot recall – possibly a Jump instruction.
D	Cannot recall – possibly a Jump instruction.
E	Unconditional Jump to the Address specified in the Addr Part (bits 1 to 12 of the Instruction).
F	Input / Output. If Addr bit 12 is set, then an Output function. If not set, then an Input function. Input – The content of the Controller (addressed by Addr bits 7 to 11) line (Addr bits 1 to 6) are transferred to the A Register. Output – The Contents of the A Register are used to specify which Bits of the Output Line (Addr bits 1 to 6) of the Controller (addressed by Addr bits 7 to 11) are turned on to energise magnets, turn on Lights etc. in the Peripheral. An **I/O Chart** will be needed to understand this.

The CPU I/O Input module is addressed by bit 11 (4.0.x), and the Output module is address as C.0.x.

OP Code bits 5 and 6 = 2.x (10)

The Addr part of the instruction is used as the location in the memory which is to be used.

Examples =

2.2.0.0.6 = Transfer the contents of the A Register into the memory location 0.0.6. The A Register is not changed.

2.4.0.0.6 = Transfer the information from memory location 0.0.6 into the A Register.

2.7.0.0.6 = Reset the Carry Bit, add the contents of memory location 0.0.6 to the contents of the A Register, with the results loaded into the A Register and the Carry Bit (if necessary).

2.0.0.0.6 = Increase the contents of memory location 0.0.6 by 1.

OP Code bits 5 and 6 = 3.x (11)

Performs almost the same as a 2.x Operation, but with an additional function.
Go to the memory location specified by the Addr bits 1 to 12, and use the information found in this memory location as the address of another memory location.

Example =

0.4.0.0.1 - Load 0.0.1 into the A register
2.2.0.0.4 – Write the 0.0.1 (contents of A Register) into memory location 0.0.4
0.4.1.2.3 – Load 1.2.3 into the A Register
3.2.0.0.4 - will load 1.2.3 (contents of the A Register) into the contents of the contents of memory location 0.0.4. So the 1.2.3 will be written into memory location 0.0.1

(3.4.x.y.z and 3.2 p.q.r are often used to transfer information from one area in memory to another area in memory.)

OP Code bits 5 and 6 = 1.x (01)

There are only two instructions 1.3.0.0.1 and 1.2.0.0.1
These two instructions are used to obtain the 6 bit Operation part of the Macro Instruction and the 12 bit Address part of the Macro instruction, and place it into the A Register.

I am unable to recall where the address of the required Macro Instruction is held when these two instructions are performed, but I would expect that the address of the required Macro Instruction would be loaded into the A Register prior to the 1.2 . . . or the 1.3 . . . instructions being performed.

The main difference between the 820/20 and the 8870, is that there were more variations of the I/O (x.F) Micro Instruction.