2.3 Program Execution

A computer follows a program stored in its memory by copying the instructions from memory into the CPU as needed. Once in the CPU, each instruction is decoded and obeyed. The order in which the instructions are fetched from memory corresponds to the order in which the instructions are stored in memory unless otherwise altered by a JUMP instruction.

To understand how the overall execution process takes place, it is necessary to consider two of the special purpose registers within the CPU: the instruction register and the program counter (see again Figure 2.4). The instruction register is used to hold the instruction being executed. The program counter contains the address of the next instruction to be executed, thereby serving as the machine's way of keeping track of where it is in the program.

The CPU performs its job by continually repeating an algorithm that guides it through a three-step process known as the machine cycle. The steps in the machine cycle are fetch, decode, and execute (Figure 2.8). During the fetch step, the CPU requests that main memory provide it with the instruction that is stored at the address indicated by the program counter. Since each instruction in our machine is two bytes long, this fetch process involves retrieving the contents of

two memory cells from main memory. The CPU places the instruction received from memory in its instruction register and then increments the program counter by two so that the counter contains the address of the next instruction stored in memory. Thus the program counter will be ready for the next fetch.

Figure 2.8 - The machine cycle

1. Retrieve the next
instruction from
memory **tas** indicated

by the program
counter) and then 2. Decode the bit pattern
increment the in the instruction register.

program counter,

3. Perform the action
required by the
Instruction in the
Instruction register,

\\

With the instruction now in the instruction register, the CPU decodes the instruction, which involves breaking the operand field into its proper components based on the instruction's op-code.

The CPU then executes the instruction by activating the appropriate circuitry to perform the requested task. For example, if the instruction is a load from memory, the CPU sends the appropriate signals to main memory, waits for main memory to send the data, and then places the data in the requested register; if the instruction is for an arithmetic operation, the CPU activates the appropriate circuitry in the arithmetic/logic unit with the correct registers as inputs and waits for the arithmetic/logic unit to compute the answer and place it in the appropriate register.

Once the instruction in the instruction register has been executed, the CPU again begins the machine cycle with the fetch step. Observe that since the program counter was incremented at the end of the previous fetch, it again provides the CPU with the correct address.

A somewhat special case is the execution of a JUMP instruction. Consider, for example, the instruction B258 (Figure 2.9), which means "JUMP to the instruction at address 58 (hexadecimal) if the contents of register 2 is the same as that of register 0." In this case, the execute step of the machine cycle begins with the comparison of registers 2 and 0. If they contain different bit patterns, the execute step terminates and the next machine cycle begins. If, however, the contents of these registers are equal, the machine places the value 58 (hexadecimal) in its program counter during the execute step. In this case, then, the next fetch step finds 58 in the program counter, so the instruction at that address will be the next instruction to be fetched and executed.

Note that if the instruction had been BO58, then the decision of whether the program counter should be changed would depend on whether the contents of register 0 was equal to that of register 0. But these are the same registers and

Figure 2.9 Decoding the instruction B258

Instruction-E B 2 5 8

/ \

Op-code B means to This f th rand is the

part of the opera!
cgange the value of address to be placed in the
the program counter program counter.

if the contents of the
indicated register is
the same as that in
register 0.

i identifies
This part of the operand iden
the register to be compared to

register 0.