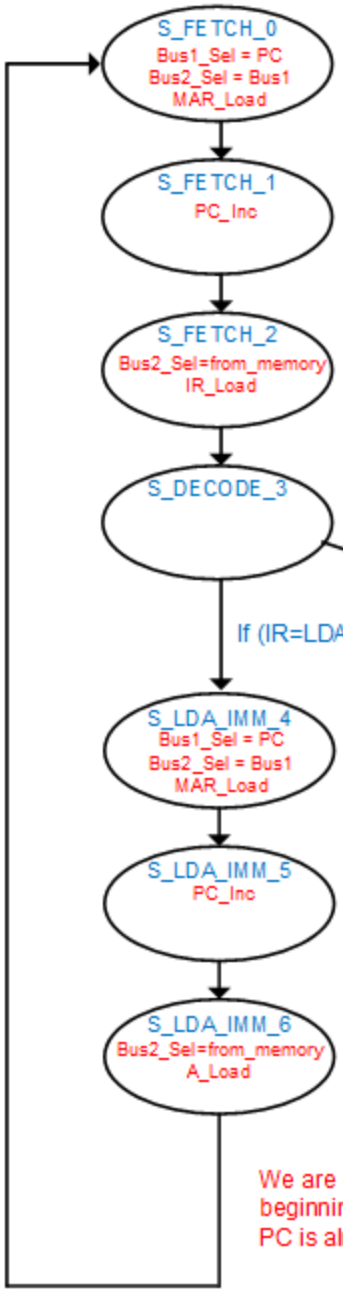
Detailed State Diagram for LDA_IMM

This load instruction moves information from memory into register A. Immediate addressing implies that information being put into A is provided as the operand of the instruction.



This state will place the PC value into the MAR in order to provide the address for the opcode. MAR will be updated with PC in the next state.

MAR is now holding the address of the opcode. It will take 1 clock cycle for the memory to provide the opcode after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The opcode that has been read from memory is now available on Bus2 and can be latched into IR by asserting IR_Load. IR will be updated with the opcode in the next state.

The opcode now resides in IR and is decoded to determine which instruction is being executed.

"Load A Immediate" means that the operand of the instruction is the information to be loaded into A. PC is already pointing to this location in memory so we can put it out on MAR. MAR will be updated with PC in the next state.

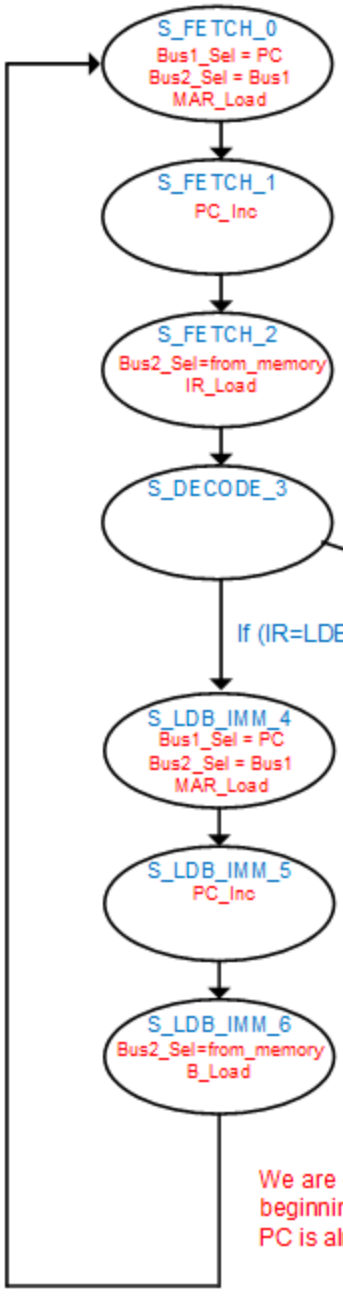
MAR is now holding the address of the operand. It will take 1 clock cycle for the memory to provide the operand after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The operand that has been read from memory is now available on Bus2 and can be latched into A by asserting A_Load. Register A will be loaded with the operand in the next state (e.g., S_FETCH_0).

We are done executing this instruction so we can return to the beginning and fetch the opcode of the next instruction. Notice that the PC is already pointing to the next address in program memory.

Detailed State Diagram for LDB_IMM

This load instruction moves information from memory into register B. Immediate addressing implies that information being put into B is provided as the operand of the instruction.



This state will place the PC value into the MAR in order to provide the address for the opcode. MAR will be updated with PC in the next state.

MAR is now holding the address of the opcode. It will take 1 clock cycle for the memory to provide the opcode after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The opcode that has been read from memory is now available on Bus2 and can be latched into IR by asserting IR_Load. IR will be updated with the opcode in the next state.

The opcode now resides in IR and is decoded to determine which instruction is being executed.

"Load B Immediate" means that the operand of the instruction is the information to be loaded into B. PC is already pointing to this location in memory so we can put it out on MAR. MAR will be updated with PC in the next state.

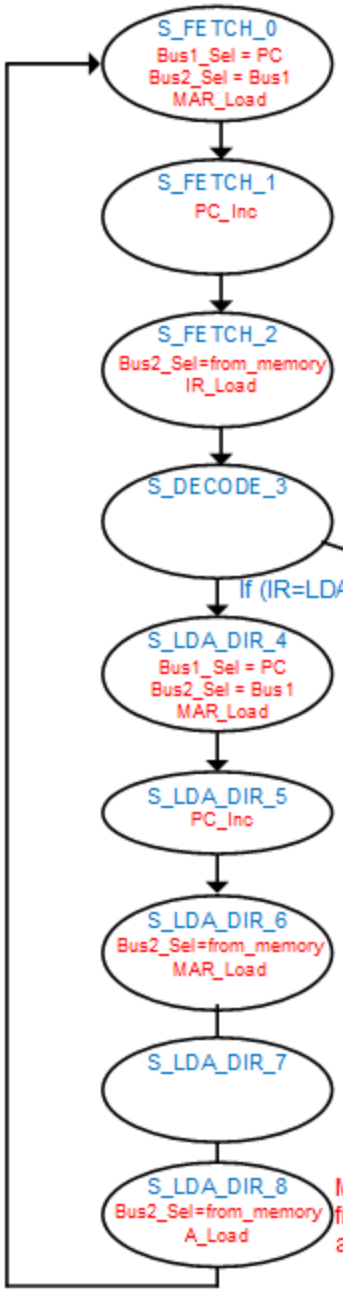
MAR is now holding the address of the operand. It will take 1 clock cycle for the memory to provide the operand after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The operand that has been read from memory is now available on Bus2 and can be latched into B by asserting B_Load. Register B will be loaded with the operand in the next state (e.g., S_FETCH_0).

We are done executing this instruction so we can return to the beginning and fetch the opcode of the next instruction. Notice that the PC is already pointing to the next address in program memory.

Detailed State Diagram for LDA_DIR

This load instruction moves information from memory into register A. Direct addressing implies that information being put into A is located at the address provided.



This state will place the PC value into the MAR in order to provide the address for the opcode. MAR will be updated with PC in the next state.

MAR is now holding the address of the opcode. It will take 1 clock cycle for the memory to provide the opcode after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The opcode that has been read from memory is now available on Bus2 and can be latched into IR by asserting IR_Load. IR will be updated with the opcode in the next state.

The opcode now resides in IR and is decoded to determine which instruction is being executed.

to other instructions....

"Load A DIrect" means that the operand of the instruction is the address of the contents to be placed in variable A. PC is already pointing to this location in memory so we can put it out on MAR.

The memory needs 1 clock cycle to provide the operand after receiving the address. During this time the PC can increment to the next address in the memory

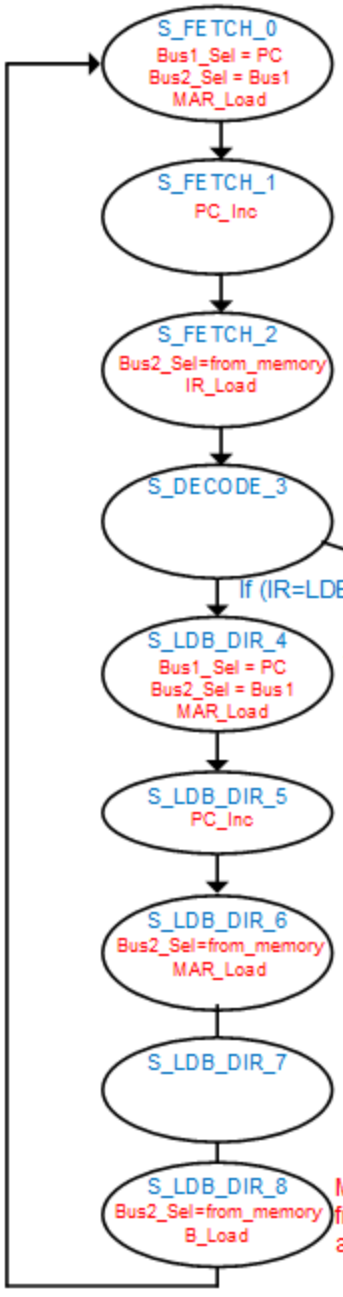
The op is now in the memory and can be picked up from Bus2 and placed into MAR by asserting MAR_load

This is an empty clock cycle that give the memory time to respond

MAR now has the correct address and the content that is on frm_mem is now placed onto bus2 and latched into place by asserting A_load

Detailed State Diagram for LDB_DIR

This load instruction moves information from memory into register B. Direct addressing implies that information being put into B is located at the address provided.



This state will place the PC value into the MAR in order to provide the address for the opcode. MAR will be updated with PC in the next state.

MAR is now holding the address of the opcode. It will take 1 clock cycle for the memory to provide the opcode after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The opcode that has been read from memory is now available on Bus2 and can be latched into IR by asserting IR_Load. IR will be updated with the opcode in the next state.

The opcode now resides in IR and is decoded to determine which instruction is being executed.

"Load B DIrect" means that the operand of the instruction is the address of the contents to be placed in variable B. PC is already pointing to this location in memory so we can put it out on MAR.

The memory needs 1 clock cycle to provide the operand after receiving the address. During this time the PC can increment to the next address in the memory

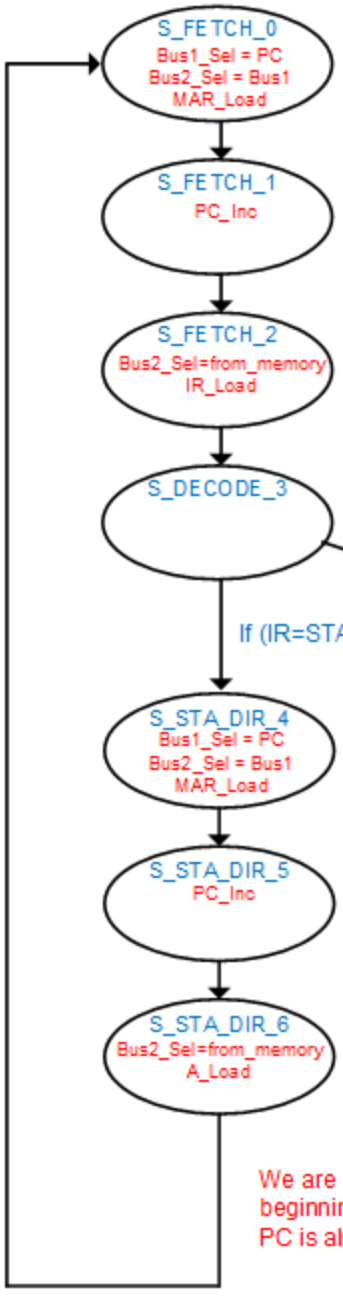
The op is now in the memory and can be picked up from Bus2 and placed into MAR by asserting MAR_load

This is an empty clock cycle that give the memory time to respond

MAR now has the correct address and the content that is on frm_mem is now placed onto bus2 and latched into place by asserting B_load

State Diagram for STA_DIR

The following is the state diagram for STA_DIR. This store instruction will move information from register A to memory. Direct addressing implies that the operand provides the address where A is to be stored.



This state will place the PC value into the MAR in order to provide the address for the opcode. MAR will be updated with PC in the next state.

MAR is now holding the address of the opcode. It will take 1 clock cycle for the memory to provide the opcode after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The opcode that has been read from memory is now available on Bus2 and can be latched into IR by asserting IR_Load. IR will be updated with the opcode in the next state.

The opcode now resides in IR and is decoded to determine which instruction is being executed.

"Store A Direct" means that the operand of the instruction is the address of where to write the contents of A. PC is already pointing to this location in memory so we can put it out on MAR. MAR will be updated with PC in the next state.

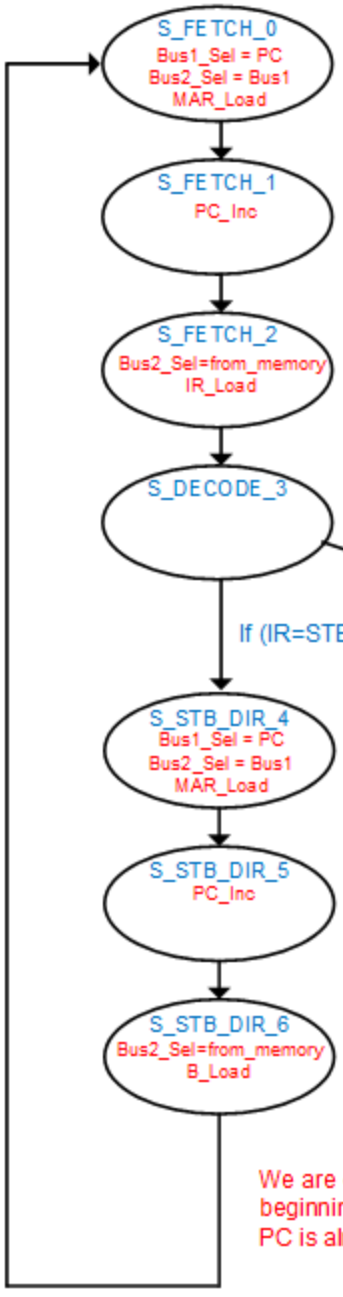
MAR is now holding the address of the operand. It will take 1 clock cycle for the memory to provide the operand after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The operand that has been read from memory is now available on Bus2 and can be latched into A by asserting A_Load. Register A will be loaded with the operand in the next state (e.g., S_FETCH_0).

We are done executing this instruction so we can return to the beginning and fetch the opcode of the next instruction. Notice that the PC is already pointing to the next address in program memory.

State Diagram for STB_DIR

The following is the state diagram for STB_DIR. This store instruction will move information from register B to memory. Direct addressing implies that the operand provides the address where A is to be stored.



This state will place the PC value into the MAR in order to provide the address for the opcode. MAR will be updated with PC in the next state.

MAR is now holding the address of the opcode. It will take 1 clock cycle for the memory to provide the opcode after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The opcode that has been read from memory is now available on Bus2 and can be latched into IR by asserting IR_Load. IR will be updated with the opcode in the next state.

The opcode now resides in IR and is decoded to determine which instruction is being executed.

"Store B Direct" means that the operand of the instruction is the address of where to write the contents of B. PC is already pointing to this location in memory so we can put it out on MAR. MAR will be updated with PC in the next state.

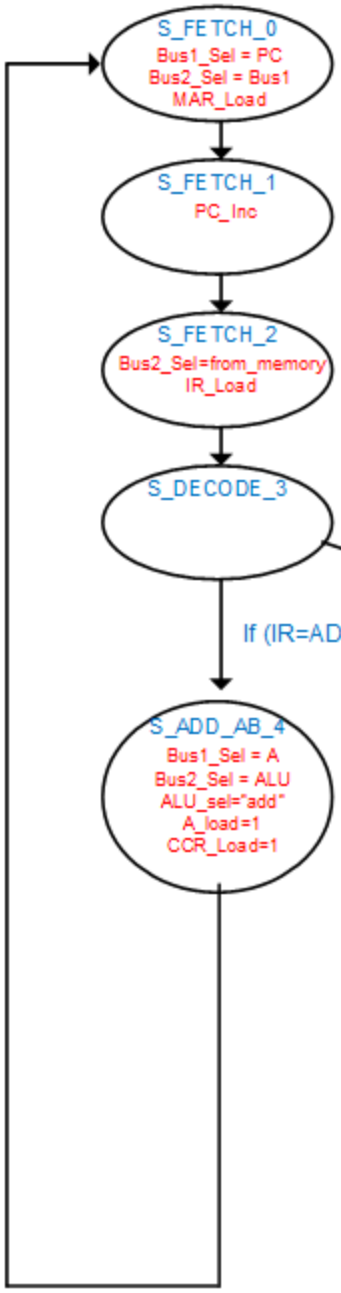
MAR is now holding the address of the operand. It will take 1 clock cycle for the memory to provide the operand after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The operand that has been read from memory is now available on Bus2 and can be latched into A by asserting B_Load. Register A will be loaded with the operand in the next state (e.g., S_FETCH_0).

We are done executing this instruction so we can return to the beginning and fetch the opcode of the next instruction. Notice that the PC is already pointing to the next address in program memory.

Example: State Diagram for ADD_AB

This instruction will use the ALU to add A and B and store the sum back in A. The status flags, NVZC, will also be generated by the ALU and latched by the condition code register



This state will place the PC value into the MAR in order to provide the address for the opcode. MAR will be updated with PC in the next state.

MAR is now holding the address of the opcode. It will take 1 clock cycle for the memory to provide the opcode after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

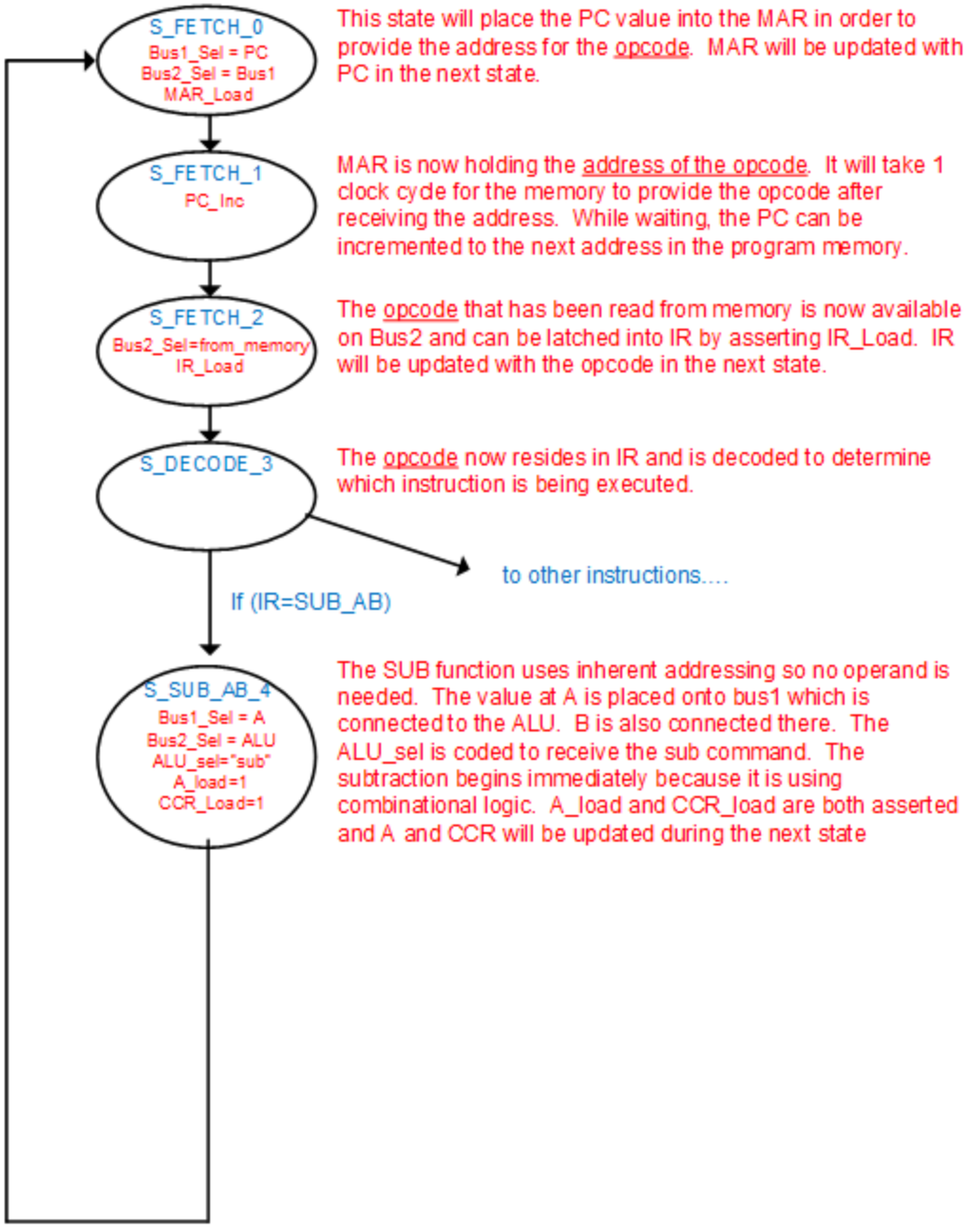
The opcode that has been read from memory is now available on Bus2 and can be latched into IR by asserting IR_Load. IR will be updated with the opcode in the next state.

The opcode now resides in IR and is decoded to determine which instruction is being executed.

The ADD function uses inherent addressing so no operand is needed. The value at A is placed onto bus1 which is connected to the ALU. B is also connected there. The ALU_sel is coded to receive the add command. The addition begins immediately because it is using combinational logic. A_load and CCR_load are both asserted and A and CCR will be updated during the next state

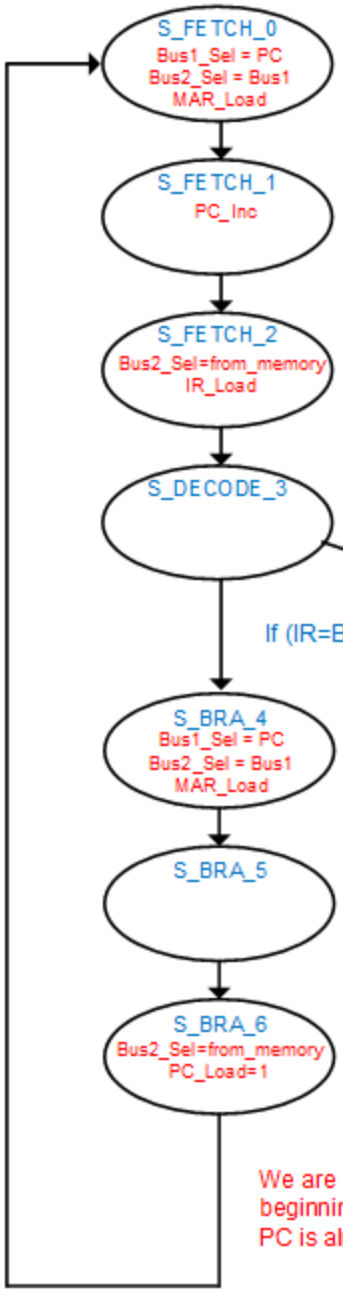
Example: State Diagram for SUB_AB

This instruction will use the ALU to subtract A and B and store the sum back in A. The status flags, NVZC, will also be generated by the ALU and latched by the condition code register



State Diagram for BRA

This load instruction will load the program counter with the address supplied by the operand of the instruction. This has the effect of setting the address of the next instruction to be executed to a new location in the program memory.



This state will place the PC value into the MAR in order to provide the address for the opcode. MAR will be updated with PC in the next state.

MAR is now holding the address of the opcode. It will take 1 clock cycle for the memory to provide the opcode after receiving the address. While waiting, the PC can be incremented to the next address in the program memory.

The opcode that has been read from memory is now available on Bus2 and can be latched into IR by asserting IR_Load. IR will be updated with the opcode in the next state.

The opcode now resides in IR and is decoded to determine which instruction is being executed.

Branch always means we are going to load the PC with the address provided by the operand. PC is placed onto MAR

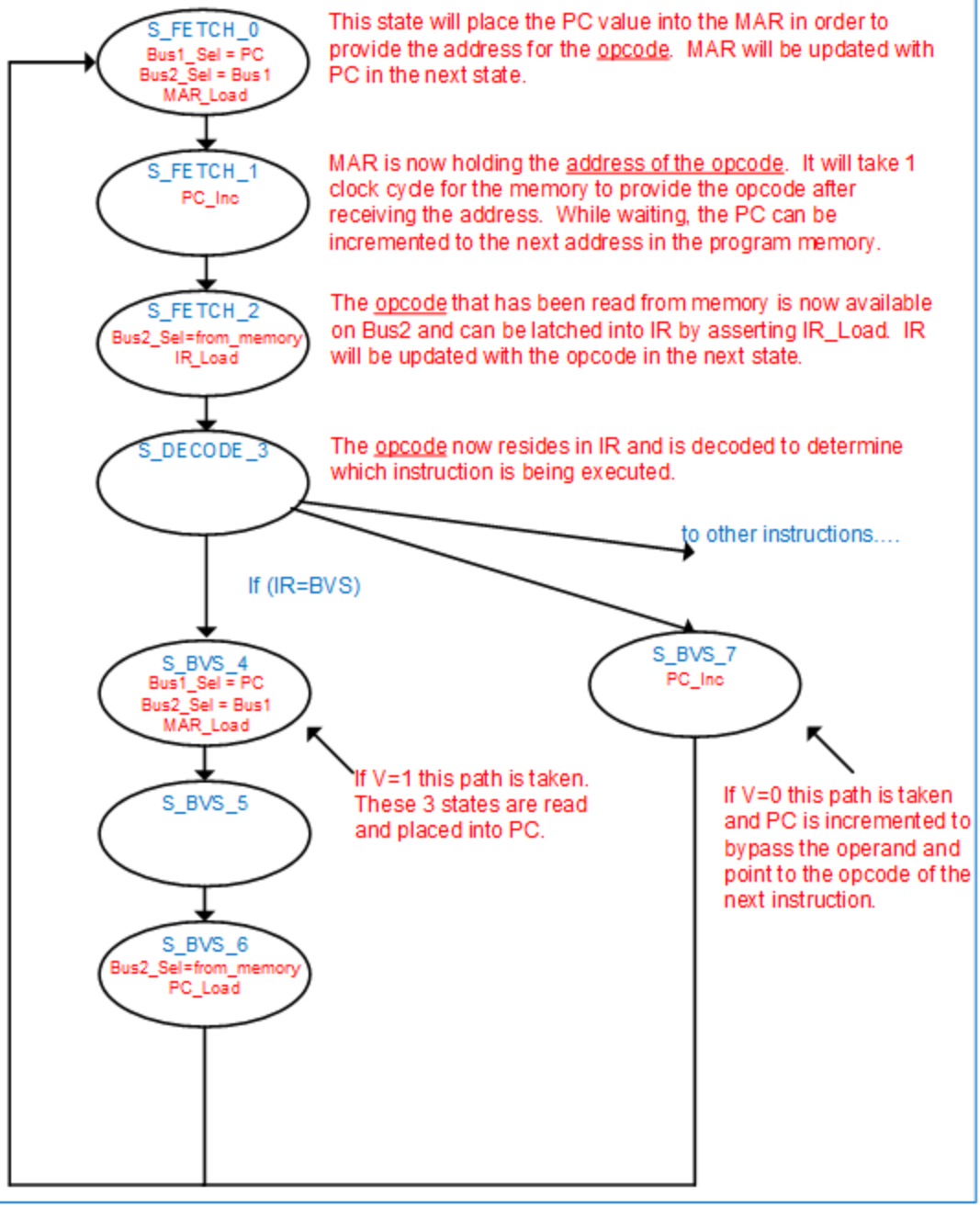
This is an empty clock cycle that allows the memory to provide the operand after receiving the address.

The operand read from memory is now available on Bus2 and can be latched into PC by asserting PC_Load

We are done executing this instruction so we can return to the beginning and fetch the opcode of the next instruction. Notice that the PC is already pointing to the next address in program memory.

State Diagram for BVS

If the overflow flag is set to V=1 this instruction will load the program counter with the address supplied by the operand. If the flag is not asserted, V=0, the branch is not taken and the program counter is incremented to the next location in memory.



State Diagram for BEQ

If the zero flag is set to =1 this instruction will load the program counter with the address supplied by the operand. If the flag is not asserted, =0, the branch is not taken and the program counter is incremented to the next location in memory.