

# Assignment 3 - XM23P Data Pipeline

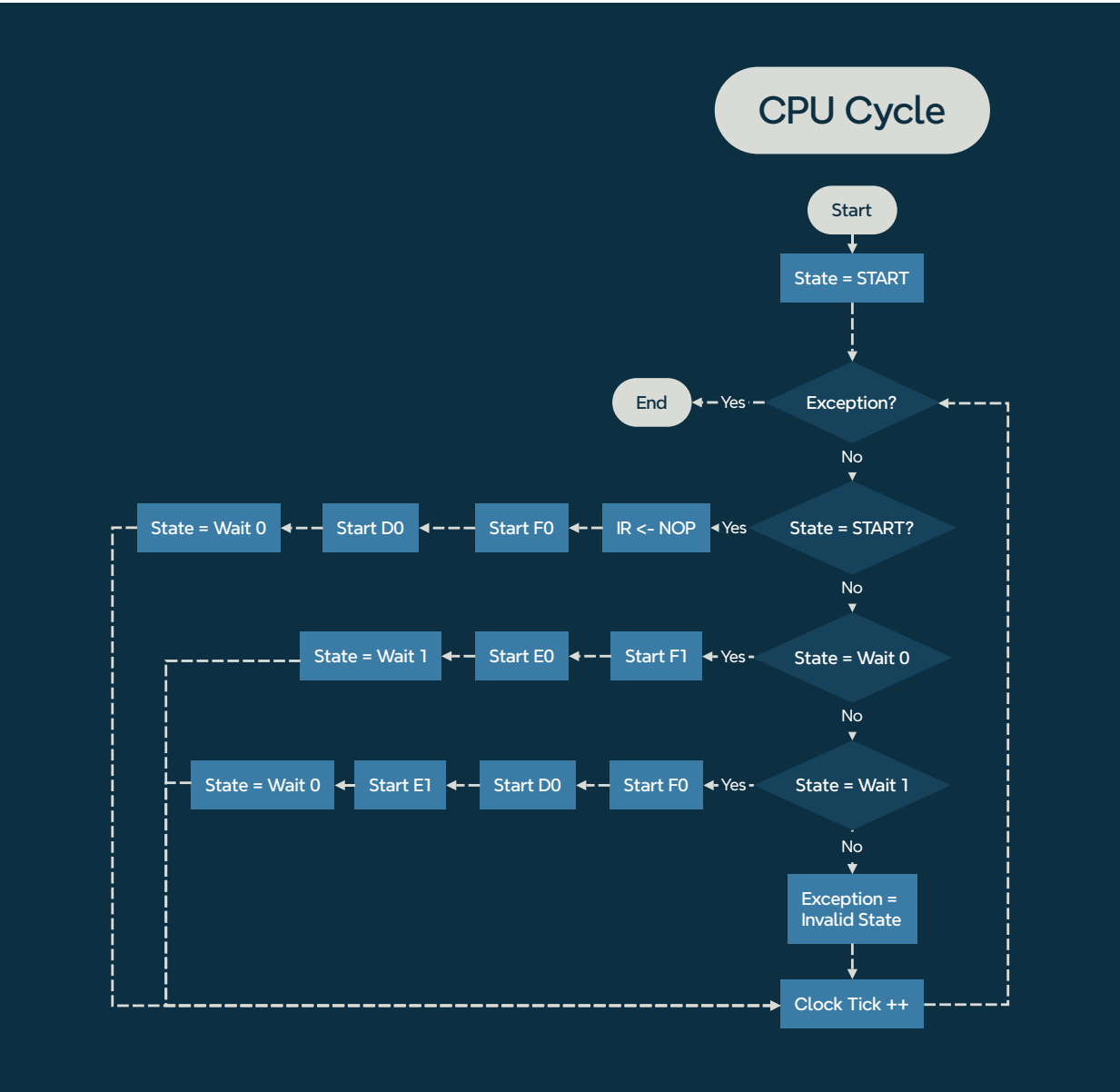
This assignment aims to implement the data memory access pipeline for the XM23P emulator. There are three stages: Fetch, Decode, and Execute. In this assignment, the Execute stage is extended to allow for data memory access in a similar manner to the instruction memory access seen in the fetch stage.

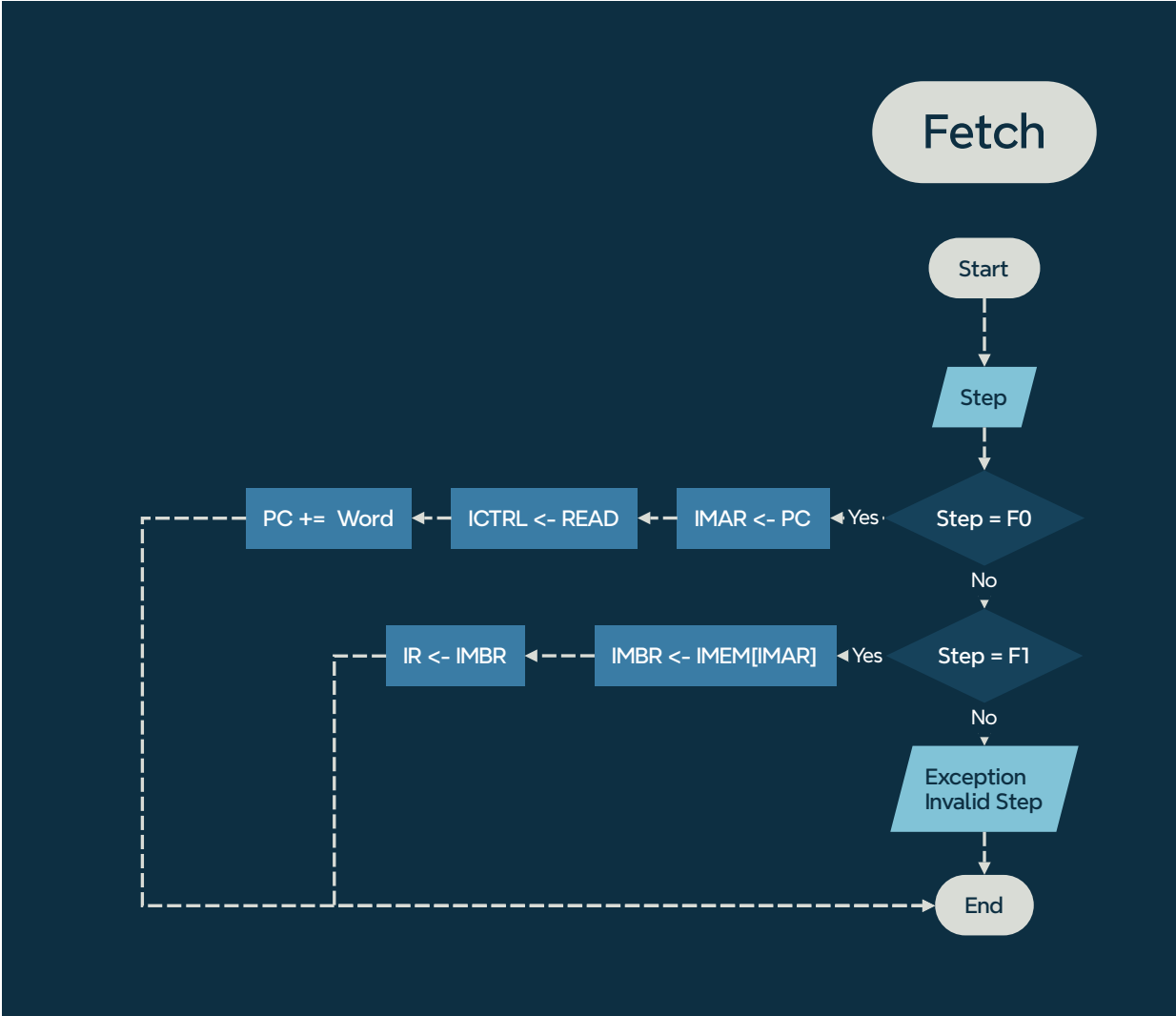
The following instructions were implemented:

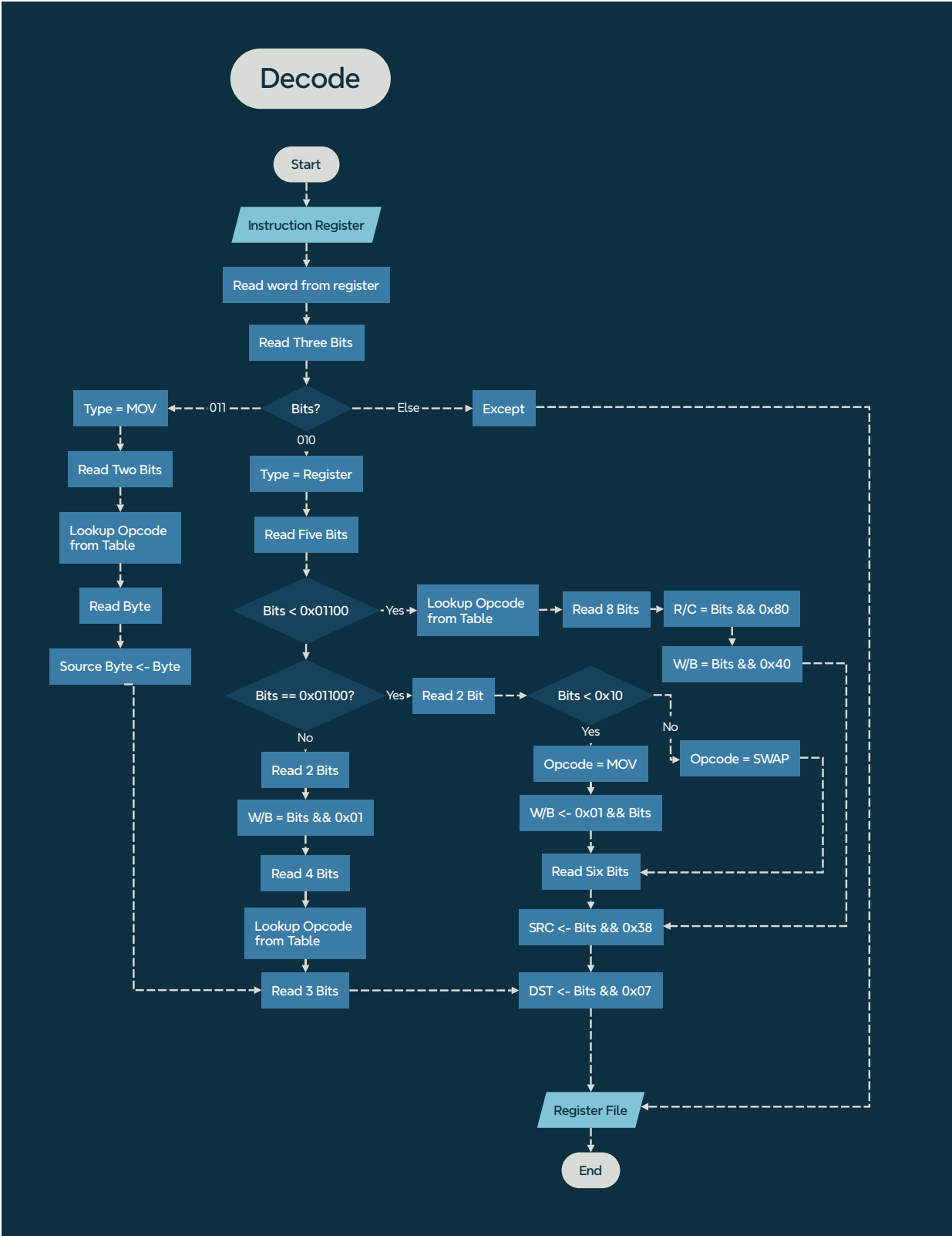
- 1. Store
- 2. Load
- 3. Store Relative
- 4. Load Relative

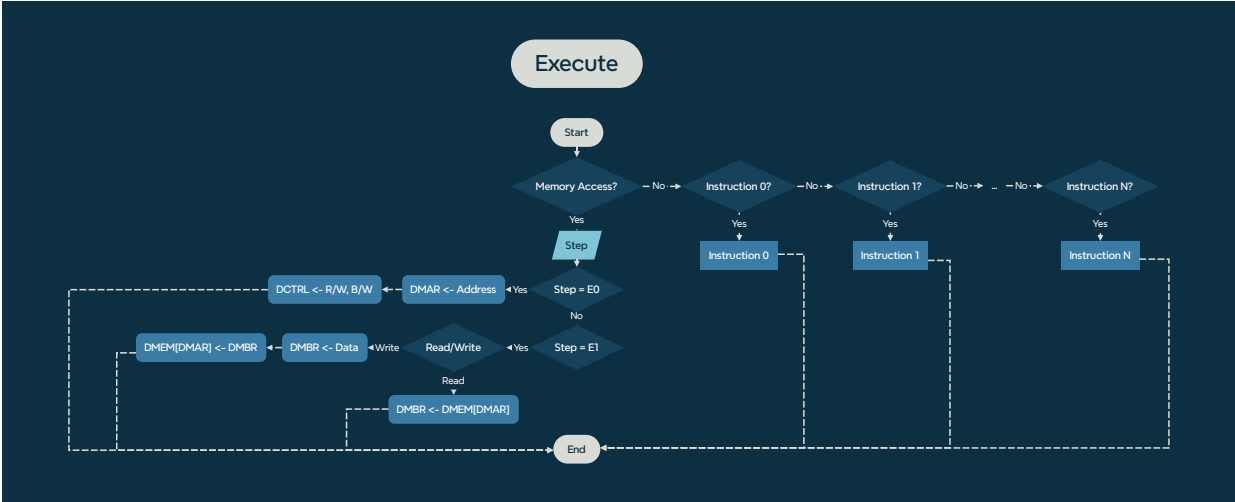
## Design

The design contains logic flowcharts detailing the CPU Loop, and the Fetch, Decode, and Execute Stages. A Data dictionary describing the instructions and register file is also included.









## Data Dictionary

IMEM	=	$32 \times 2^{10} \{\text{WORD}\}$
IMAR	=	ADDRESS
ICTRL	=	[READ WRITE]
IMBR	=	WORD
IR	=	WORD
DMEM	=	$64 \times 2^{10} \{\text{BYTE}\}$
DMAR	=	ADDRESS
DCTRL	=	[READ WRITE]
DMBR	=	WORD
REGFILE	=	$3\{\text{GPR}\}3 + \text{BP} + \text{LR} + \text{SP} + \text{PC}$
GPR	=	WORD *General Purpose Register*
BP	=	WORD *Base Pointer*
LR	=	WORD *Link Register*
SP	=	WORD *Stack Pointer*
PC	=	WORD *Program Counter*
PSW	=	$\text{PRV\_PRI} + 4\{\text{DC}\}4 + \text{FLT} + \text{CUR\_PRI} + \text{V} + \text{SLP} + \text{N} + \text{Z} + \text{C}$
PRV_PRI	=	$3\{\text{BIT}\}3$ *Previous Priority*
DC	=	BIT *Don't Care*
FLT	=	BIT *Fault*
CUR_PRI	=	$3\{\text{BIT}\}3$ *Current Priority*
V	=	BIT *Arithmetic overflow*
SLP	=	BIT *Sleep State*
N	=	BIT *Negative Result*
Z	=	BIT *Zero Result*
C	=	BIT *Carry*
BREAKPOINT	=	ADDRESS
START_ADDRESS	=	ADDRESS
INSTRUCTION	=	$\text{CODE} + 1\{\text{PARAMETER}\}4$
CODE	=	[0-20] *Contiguous encoding of instructions*
PARAMETER	=	[RC WB SOURCE DESTINATION BYTE]
RC	=	BIT
WB	=	BIT
SOURCE	=	$3\{\text{BIT}\}3$
DESTINATION	=	$3\{\text{BIT}\}3$
READ	=	0x0000
WRITE	=	0x0001
ADDRESS	=	WORD
WORD	=	$2\{\text{BYTE}\}2$
BYTE	=	$8\{\text{BIT}\}8$
BIT	=	[0 1]

## Testing

---

The following tests were implemented:

- Test\_28: Store Instructions
- Test\_29: Load Instructions
- Test\_30: Data Hazard Bubble
- Test\_31: Negative Offsets
- Test\_32: Execute Pipeline

Each test may be run from a powershell terminal with the following command:

```
Get-Content '.\Path\To\Input\File' | '.\Path\To\Executable'
```

## Test\_28: Store Instructions

### Purpose

Verify store and store relative instructions correctly add data to data memory.

### Configuration

.\tests\Execute\_Tests\Input\_Files\Test28.in

1. Test28\_Store.xme was loaded into the emulator.
2. **b 112** was entered to set a breakpoint at address **#0112**
3. **g** was entered to run the program
4. **m 1 1000 1020** was entered to dump the data memory between address **1000** and **1020**.
5. **r** was entered to dump the CPU register contents.

### Expected Results

**#00CE** should be at data memory addresses **#1000**.

**#FACE** should be stored at data memory address **#1010**. **#1001** should be stored in Register **R0**.

### Results

The memory and register contents correctly matched:

```
#1000  00 ce 00 00 00 00 00 00 00 00 00 00 00 00 00
#1010  ce fa 00 00 00 00 00 00 00 00 00 00 00 00 00

User> Register Dump Utility
R0: 1001
R1: face
R2: 0000
R3: 0000
R4: 0000
R5: 0000
R6: 0000
R7: 0112
```

### Pass/Fail

Pass.

## Test\_29: Load Instructions

### Purpose

Verify load and load relative instructions correctly retrieve data from data memory.

### Configuration

.\tests\Execute\_Tests\Input\_Files\Test29.in

1. Test29\_Load.xme was loaded into the emulator.
2. **b 120** was entered to set a breakpoint at address **#0120**
3. **g** was entered to run the program
4. **m 1 1000 1020** was entered to dump the data memory between address **1000** and **1020**.
5. **r** was entered to dump the CPU register contents.

### Expected Results

The CPU registers should be as follows:

```
R0: 1000
R1: FACE
R2: FACE
R3: 00CE
R4: 0000
R5: 0000
R6: 0000
R7: 0120
```

**#FACE** should be stored at **#1002**

**#00CE** should be stored at **#1012**

### Results

The register and memory contents correctly matched:

```
#1000  00 00 ce fa 00 00 00 00 00 00 00 00 00 00 00 00
#1010  00 00 ce 00 00 00 00 00 00 00 00 00 00 00 00

User> Register Dump Utility
R0: 1000
R1: face
R2: face
R3: 00ce
R4: 0000
R5: 0000
R6: 0000
R7: 0120
```

### Pass/Fail

Pass.



## Test\_30: Data Hazard Bubble

### Purpose

Verify that a data hazard bubble is inserted when a load instruction changes the program counter.

### Configuration

.\tests\Execute\_Tests\Input\_Files\Test30.in

1. Test30\_DataHazard.xme was loaded into the emulator.
2. **b 202** was entered to set a breakpoint at address **#0202**
3. **g** was entered to run the program
4. **r** was entered to dump the CPU register contents.

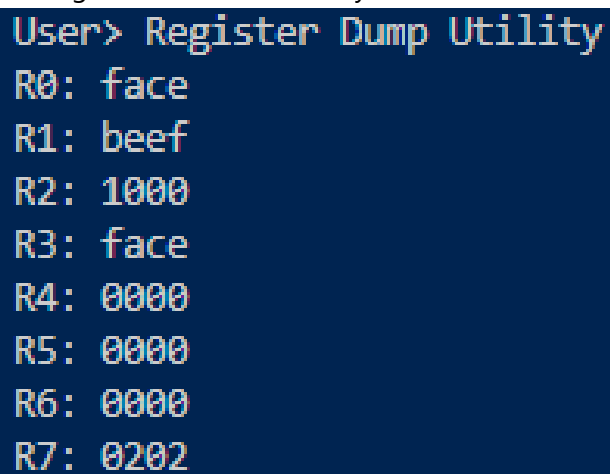
### Expected Results

The CPU registers should be as follows:

```
R0: FACE
R1: BEEF
R2: 1000
R3: FACE
R4: 0000
R5: 0000
R6: 0000
R7: 0202
```

### Results

The register contents correctly matched:



```
User> Register Dump Utility
R0: face
R1: beef
R2: 1000
R3: face
R4: 0000
R5: 0000
R6: 0000
R7: 0202
```

### Pass/Fail

Pass.

## Test\_31: Negative Offsets

### Purpose

Verify that negative offsets are correctly handled in load and store relative instructions.

### Configuration

.\tests\Execute\_Tests\Input\_Files\Test31.in

1. Test31\_NegativeOffsets.xme was loaded into the emulator.
2. **b 10E** was entered to set a breakpoint at address **#010E**
3. **g** was entered to run the program
4. **m 1 0FC0 0FD0** was entered to dump the data memory between address **0FC0** and **0FD0**.
5. **r** was entered to dump the CPU register contents.

### Expected Results

The value **#FACE** should be loaded into Register **R3**.

The value **#FACE** should be stored at data memory address **#0FC0**

### Results

The register and memory contents correctly matched:

```
#0fc0  ce fa 00 00 00 00 00 00 00 00 00 00 00 00 00 00
User> Register Dump Utility
R0: face
R1: 0000
R2: 1000
R3: face
R4: 0000
R5: 0000
R6: 0000
R7: 010e
```

### Pass/Fail

Pass.

Test\_32: Execute 1 Pipeline

Purpose

Verify that the pipeline executes E1 stage only for data memory accesses.

Configuration

.\tests\Execute\_Tests\Input\_Files\Test32.in

- 1. Test32\_Execute1Pipeline.xme was loaded into the emulator.
- 2. **b 10A** was entered to set a breakpoint at address **#010A**
- 3. **d** was entered to enable debug mode
- 4. **g** was entered to run the program

Expected Results

Instructions at PC = 100 and 106 should not have an E1 stage.

Instructions at PC = 102 and 104 should have an E1 stage.

Results

The pipeline debug statements showed the correct stages for each instruction:

Clock	PC	Instruction	Fetch	Decode	Execute
0000	0102	0000	F0: 0100	D0: 0000	
0001	0102	4001	F1: 4001		E0: 0000
0002	0104	4001	F0: 0102	D0: 4001	
0003	0104	e002	F1: e002		E0: 4001
0004	0106	e002	F0: 0104	D0: e002	
0005	0106	a013	F1: a013		E0: e002
0006	0108	a013	F0: 0106	D0: a013	E1: e002
0007	0108	420a	F1: 420a		E0: a013
0008	010a	420a	F0: 0108	D0: 420a	E1: a013
0009	010a	0000	F1: 0000		E0: 420a
0010	010c	0000	F0: 010a	D0: 0000	

Breakpoint Reached. CNVZ: 0001

Pass/Fail

Pass.