



MODULAR PROGRAMME
ASSESSED COURSE-WORK SPECIFICATION

Module Details:

Module Code: UFS004C2	Module Title: CPU Architecture & VHDL	
Module Leader:	Nigel Gunton	
Module Tutors: Nigel Gunton Rob Williams		
Assignment CW1	Element Number: Weighting 25%	Total Assignment Time: 12 hrs

Dates:

Date assignment issued to students: wb Oct 27th '03	Date for return of marked work: wb Feb. 2nd '04
Submission Place: post-box in N foyer, below the North stairs	Date of Submission: wb 15th Dec. 03
	Time of Submission: 10.00am

Deliverables:

<p>As listed on the Assignment spec sheet</p>

Foreword

This assignment is the first part of a two part assignment. Work done for this assignment will be extended during the second assignment. The minimum hand-in requirement for this phase is indicated below, along with the required demonstrations and discussions. Phase two will include the integration of the ALU and the register set with the control unit to form a complete CPU. This design will then be extended by the addition of new instructions and registers.

You may work in pairs for this assignment.

Overview

Modern 'systems on silicon' may often require a complex control unit or sequencer. This can have similarities to the control unit of a simple CPU in that it has to generate a sequence of control signals dependent on the current input condition and current 'instruction'. It is also possible that a design for a system is best implemented by a simple CPU like device without requiring the complexity of a third party CPU core as 'bought in' IP, or incurring the costs of devices with a CPU core already embedded in them.

To better understand this design process, you are required to develop the control unit for the 'Carpinelli RS-CPU'¹. Control units can be divided into 2 types, 'hardwired' and micro-sequence. For the purposes of this assignment you will be developing a hardwired control unit.

The design process will be supported by the lectures and support will be given in the lab sessions. However the design process is summarized here.

- Develop the state diagram for the fetch and decode cycles for the CPU. This will be represented as a series of branches from the last state of the FETCH sequence to the individual execute routines. Note that more than one path of execution will be required for those instructions that are conditional.
- Expand the execute sequences to the required number of steps to create a complete state diagram for the CPU
- Establish the data transfers and data paths needed and regroup the data transfers according to the register that is affected by the transfer.
- Identify the enable, control and load signals that are required for control and develop the final register model for the CPU.
- Develop a mapping between the Op-Codes and the state value.
- Implement the control unit in VHDL and test under simulation for correct execution of the instruction set.

Deliverables

- 1) A complete state diagram for the RS-CPU, this should show the trigger events for transitions and state names only. This may be in ASM form or as a state transition diagram (circles and arcs).

DIAGRAM

- 5 marks

- 2) The RTL for all states. This should be provided as
 - i) a table showing execution for each op-code and
 - ii) a table for each target register

RTL TABLE i)

- 10 marks

RTL TABLE ii)

- 10 marks

¹ From John Carpinelli's design for a relatively simple CPU

- 3) A diagram showing the final register section for the RS-CPU. This should show bus widths, control signals, buffers, ALU, zero flag etc.

DIAGRAM - 5 marks

- 4) Your control unit design. This should clearly show how the control unit generates the control signals. It may contain both written explanation and schematic representations. You must provide a 'walk-through' of your design with your lab tutor.

CU DESIGN - 15 marks

Walk-through - 5 marks

- 5) A VHDL implementation of your design. This should use the data-flow subset of the language to describe the decoders.

VHDL CODE - 20 marks

Test Results - 10 marks

Demonstration - 20 marks

RS-CPU Specification

The following diagrams and tables provide the design details of the CPU and the first stages of the development process for the hardwired control unit. The ALU design and its control signals are also provided.

This CPU can address 64 Kbytes of memory, each word is 8 bits wide. This is via address pins A[15..0] and bidirectional data pins D[7..0]. There are three programmer accessible registers :

AC The accumulator, AC, is used to receive the results of all arithmetic and logical operations that require 2 operands. It also provides one of the operands for these operations. It is also used for all data transfers between the CPU and memory.

R Register R is an 8-bit general purpose register that provides the second operand for all 2 operand arithmetic and logical operations. It can also be used for the temporary storage of data.

Z A one-bit zero flag, Z, is modified whenever an arithmetic or logical operation is performed.

The CPU also contains the following registers

AR 16-bit address register, AR, supplies an address to memory via A[15..0]

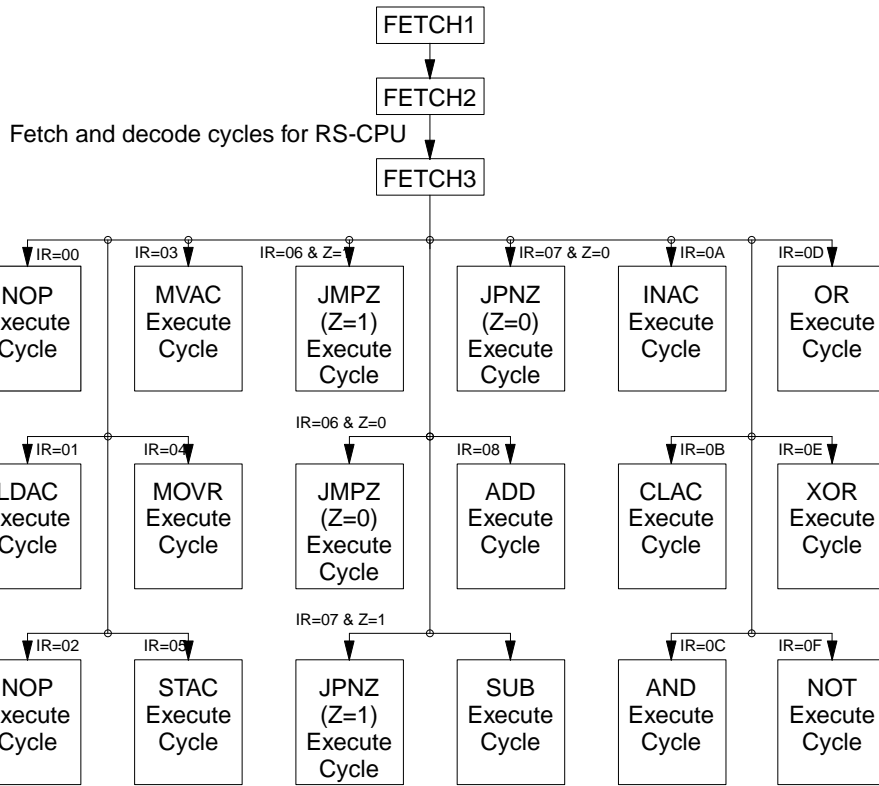
PC 16-bit program counter, PC, contains the address of the next instruction to be executed, or the address of the next required operand of the instruction.

DR 8-bit data register, DR, receives instructions and data from memory and transfers data to memory via D[7..0].

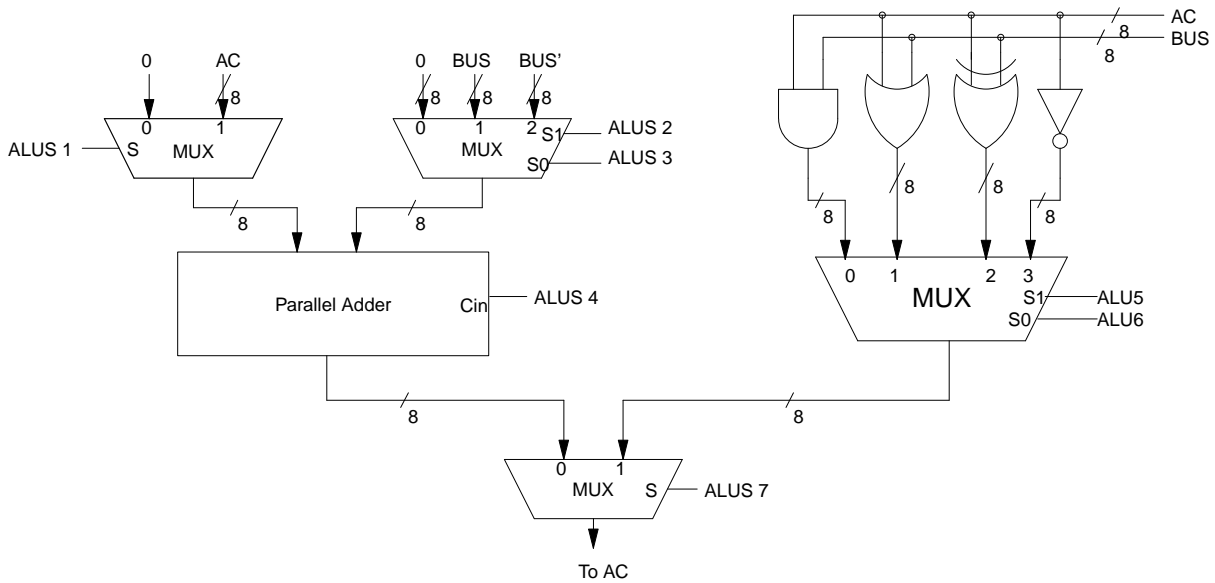
TR 8-bit temporary register, TR, stores data during instruction execution, eg. to hold the low byte of operand addresses.

Note that the program counter can hold the address of the next operand as well as the address of the next instruction. In fact it is not known whether the address is that of an instruction or an operand until the current instruction is decoded.

All arithmetic and logic is done on the contents of register R and the accumulator for 2 operand operations and the accumulator only for single operand operations.



ALU implementation for the RS-CPU



Instruction set for a relatively simple CPU		
Instruction	Instruction Code	Operation
NOP	0000 0000	No operation
LDAC	0000 0001 Γ	$AC \leftarrow M[\Gamma]$
STAC	0000 0010 Γ	$M[\Gamma] \leftarrow AC$
MVAC	0000 0011	$R \leftarrow AC$
MOVR	0000 0100	$AC \leftarrow R$
JUMP	00000101 Γ	GOTO Γ
JMPZ	0000 0110 Γ	IF (Z=1) THEN GOTO Γ
JPNZ	0000 0111 Γ	IF (Z=0) THEN GOTO Γ
ADD	0000 1000	$AC \leftarrow AC + R$, IF (AC + R = 0) THEN Z \leftarrow 1 ELSE Z \leftarrow 0
SUB	0000 1001	$AC \leftarrow AC - R$, IF (AC - R = 0) THEN Z \leftarrow 1 ELSE Z \leftarrow 0
INAC	0000 0101	$AC \leftarrow AC + 1$, IF (AC + 1 = 0) THEN Z \leftarrow 1 ELSE Z \leftarrow 0
CLAC	0000 1011	$AC \leftarrow 0$, Z \leftarrow 1
AND	0000 1100	$AC \leftarrow AC \wedge R$, IF (AC \wedge R = 0) THEN Z \leftarrow 1 ELSE Z \leftarrow 0
OR	00001101	$AC \leftarrow AC \vee R$, IF (AC \vee R = 0) THEN Z \leftarrow 1 ELSE Z \leftarrow 0
XOR	0000 1110	$AC \leftarrow AC \oplus R$, IF (AC \oplus R = 0) THEN Z \leftarrow 1 ELSE Z \leftarrow 0
NOT	0000 1111	$AC \leftarrow AC'$, IF (AC' = 0) THEN Z \leftarrow 1 ELSE Z \leftarrow 0

