WORKSHOP ON COMPUTER PROCESSING & TRANSMISSION OF THE ARABIC LANGUAGE

KUWAIT, 14-16 APRIL, 1985

A COMPUTER ARCHITECTURE TO SUPPORT ARABIC TEXT PROCESSING

MAMDOUH M.S. NAJJAR and MANZER MASUD

Sponsoring Agencies:

- The Economic Commission for Western Asia (ECWA)
- Kuwait Institute for Scientific Research (KISR)
- The Arab Fund for Economic and Social Development (AFESD)

A COMPUTER ARCHITECTURE TO SUPPORT ARABIC TEXT PROCESSING

prepered for

Workshop on Computer Processing and transmission of the Arabic Language

by

Mamdouh M-S. Najjar and Manzer Masud
Computer Science and Engineering Department
University of Petroleum and Minerals
Dhahran, Saudi Arabia

ABSTRACT

Most of the 'Arabic computers' in use today are based on standard CPUs which are attached to especially designed I/O devices to handle Arabic text. In this paper we presents the architecture of a CPU especially designed to support Arabic text processing. In contrast to standard CPUs the machine described here has the capability to handle and manipulate character shapes and vowels. The advantages are : reduced overall system cost, improved performance and throughput, and ease of programming.

JANUARY 1985

1. INTRODUCTION

The use of computers in the Arabic-speaking countries is ever increasing. In order to harness the full potential of the technology, it is necessary to establish good communication between the machine and its Arabic-speaking users. Today, this is being done by means of specially designed Input/Output devices. The approach is to bulid a special processor within the I/O device which performs textual analysis to determine the shape of the character at the time of output [4]. This results in a high cost of Arabic I/O devices. Since one CPU may be connected to several such devices the overall system cost increases considerably.

In this paper we describe a special purpose computer architecture to support Arabic text processing. It will be possible to interface the machine to any standard I/O device with modified character generator ROM for Arabic output. The advantages will be as follows.

1- Overall system cost will reduce because special purpose intelligent I/O devices will not be needed.

2- System Performance and throughput will improve.

3- The system will be more flexible in handling Arabic characters.

2. OVERVIEW OF THE PROBLEM

An Arabic letter may have one of the four possible shapes depending on the linking characteristics of characters adjacent to it [4]. Intelligent devices are available on the market which output appropriate shape of a letter by examining characters adjacent to it. Since the devices must do some processing to select an appropriate character shape they are slower and more expensive than their Latin counterparts. Text processing of vowelized (Harakat) Arabic is even more difficult. Although vowels are not explicitly written in common day-to-day text they are essential for religious and legal writings. For This reason Arabic office automations system or systems supporting Arabic data bases must be capable of handling vowelized text in a convenient way. The current approach of storing vowels as a separate character is, we believe, inflexible and wasteful. Furthermore, it makes the output devices even more expensive and slower. To our knowledge no present day output device can handle vowelized text to the user's satisfaction.

One way to reduce the complexity of output devices is to build a special-purpose CPU to support Arabic text processing. This can be done by including special instructions which will allow the programmer to manipulate character shapes and vowels. In section 4 we propose one such machine and its complete instruction set as well as other architectural features. The user of the machine will have an option of storing characters with appropriate shape and vowels, if desired, at the time of the input. The shape of the character will be determined by the input program based on the linking characteristics of adjacent characters. Storing character shapes will allow more efficient searching as well as faster output and may improve system performance in certain applications. The user will also have the option of storing characters without regard to its shapes or vowels. In which case the output routine will provide character shapes, or an existing intelligent output device may be used.

3. A 14-BIT ARABIC CODE

In order to achieve the objective of processing Arabic text including character shape and vowels we propose a new 14-bit Arabic character code as depicted in Figure 1. The code consists of three fields, and may be interpreted as concatenation of three sub-codes. The first 7 bits are for the standard Arabic character code, the next two bits are for shape, and the last 5 bits are for vowels.

The 7-bit character code will follow the standards proposed in [5]. With the 2-bit shape code we may define one of the 4 possible shapes of a character as shown in Table 1. The 5-bit vowel code is sufficient to encode the 5 basic vowels as well as their allowable combinations. In addition certain special punctuation marks may also be encoded. Table 2 describes the encoding in details.

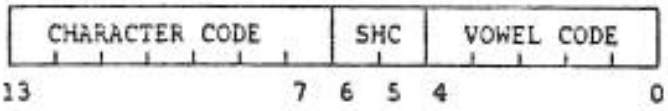

Figure 1. Arabic 14-bit code.

S1	S2	Connection
0	0	Isolated
0	1	Right
1	0	Left
1	1	Both sides

Table 1. Shape code.

			V4 V5			
V1	V2	V3	00	01	10	11
0	0	0	null	--	--	--
0	0	1	--	--	--	--
0	1	0	--	--	--	--
0	1	1	--	--	--	--
1	0	0	--	--	--	--
1	0	1	--	--	--	--
1	1	0	--	--	--	--
1	1	1	--	--	--	--

Table 2. Vowels code.

4. *MACHINE ORGANIZATION AND DESIGN*

Our major objective is to demonstrate the feasibility and usefulness of building a special-purpose computer to support Arabic text processing. However, the machine must also support general-purpose computing tasks and must include instructions for data moving, arithmetic and logic processing, and program control. We chose the PDP-11, perhaps the most popular minicomputer, as the starting point in our design. The machine has a regular and proven architecture and proved to be a good choice. In order to simplify the implementation, less frequently used instructions as well as some of the addressing

modes were eliminated. The reduced machine was then augmented with instructions especially designed for Arabic text processing. An overview of the design is given below. The 16-bit architecture can easily be expanded to 32 bits if a more powerful version is desired.

4.1 Basic bus structure

The basic bus structure of the machine is shown in Figure 2. The CPU can communicate with the I/O devices through the I/O Bus, and communicate with memory through Memory Bus. This approach is different from the PDP-11 UNIBUS. The prime motivation was to increase I/O throughput. The priority memory controller (PMC) will control the memory access requests from CPU or from direct memory access (DMA) controller.

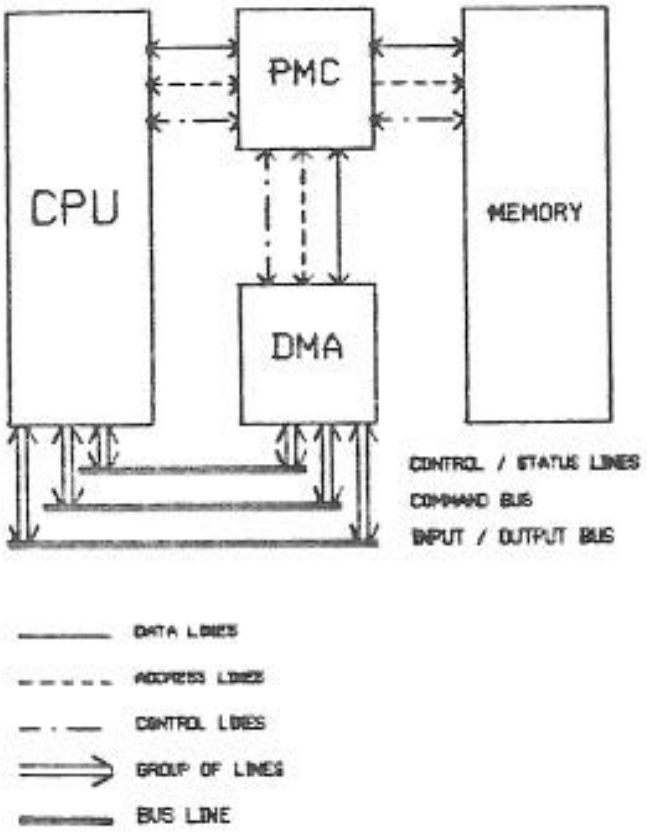

FIGURE 2. BUS STRUCTURE

4.2 Layout of the machine

The machine is a 16-bit word length computer. It has 16 general purpose registers of 16-bit each. The names R0, R1, ... , R15 will be used as general purpose registers. R15 is the program counter (PC), and R14 is the stack pointer (SP).

The memory is byte addressable and the total address space is 64K bytes or 32K words. Like PDP-11, words must start at even addresses. The address space can be increased beyond 64K by including external memory management modules.

4.3 Arabic character representation

The machine will support 3 different Arabic character representations depending on the I/O devices and the programmer's options.

1- The full character representation is shown in Figure 3. This representation is based on the new Arabic 14-bit code discussed earlier. Bits 15 and 7 of the word are not used in this code. They can be set to zero at input time. Bits 8 through 14 are for the character, bits 5 and 6 are for the shape code (SHC) and bits 0 through 4 are for vowel code (VC). This code will allow the programmer to manipulate the shape and vowels of Arabic characters by means of instructions discussed in 4.5.3 .

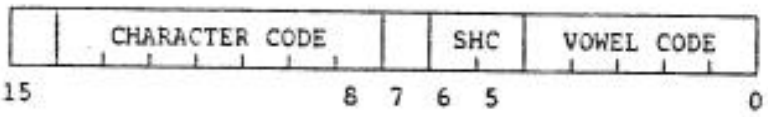

Figure 3. Full character representation.

2- The short character (7 bits) representation is shown in Figure 4. This code is useful when the user prefers to use Arabic text without regard to its shape and vowels. Using this code two characters may be packed in one word. Bits 15 and 7 are not used in this code and can be set to zero at input. If this code is used at input then it will be necessary to provide the shape code at the time of the output or to use intelligent output devices which have the capability to determine character shape.

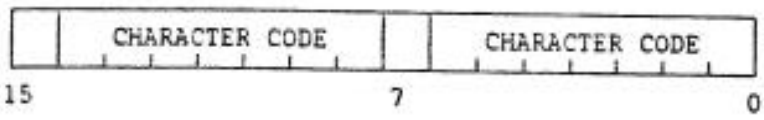

Figure 4. Short character (7 bits) representation.

3- short character (8 bits) representation is shown in Figure 5. This code is similar to the previous one except that the character code is 8 bits long.

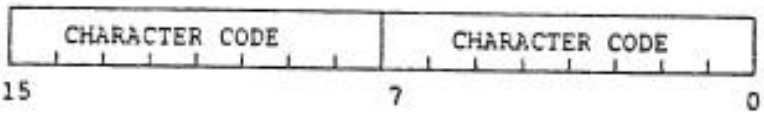

Figure 5. Short character (8 bits) representation.

4.4 *Addressing modes*

Out of the eight addressing modes available in the PDP-11 minicomputer ,we selected only 4. This was necessary to simplify the implementation as well as to increase the number of working registers to 16. The selection was based on the frequency of usage of these modes. Table 3 list the four modes. A detaild description may be found in [2, 6] The address field for either source or destination is shown in Figure 6. The addressing mode field (AM) is two bits long crossponds to four addressing modes. The register field (Reg) is four bits long crossponds to 16 general purpose registers.

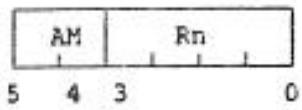

Figure 6. Address field.

NAME	CODE	SYNTAX	MEANING
Register	0 0	Rn	Ea ← Rn
Register indirect	0 1	@Rn	Ea ← [Rn]
Autoincrement	1 0	[Rn]+	Ea ← [Rn]; Increment Rn
Autodecrement	1 1	-[Rn]	Decrement Rn; Ea ← [Rn]

Table 3. Addressing modes.

4.5 *Instruction Set*

The basic instruction set of the machine include the most frequently used instructions in PDP-11 minicomputer. A detailed description of each type is described in the following sections.

4.5.1 *Two-operand instructions*

There are 14 two-operand instructions listed in Table 4. These instructions allow complete freedom in specifying the source and destination operands using any of the four addressing modes. Their format is shown in Figure 7.

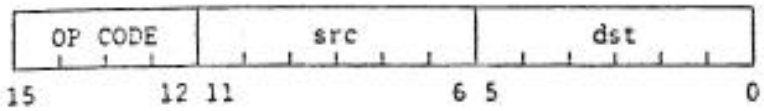

Figure 7. Two operand instruction format.

Mnemonic	Meaning	OP code	operation performed
ADD	add	0 0 0 1	dst ← [src] + [dst]
SUB	subtract	0 0 1 0	dst ← [src] - [dst]
AND	logical AND	1 1 0 0	dst ← [src] AND [dst]
OR	logical OR	1 1 0 1	dst ← [src] OR [dst]
XOR	logical XOR	1 1 1 0	dst ← [src] XOR [dst]
BIC	bit clear	1 1 1 1	dst ← $\overline{[src]}$ AND [dst]
MOV	move word	1 0 0 0	dst ← [src]
MOVB	move byte	1 0 0 1	dst ← [src]
MOVS	move shape	1 0 1 0	dst ← [src]
MOVV	move vowel	1 0 1 1	dst ← [src]
CMP	compare word	0 1 0 0	[src] - [dst]
CMPB	compare byte	0 1 0 1	[src] - [dst]
CMPS	compare shape	0 1 1 0	[src] - [dst]
CMPV	compare vowel	0 1 1 1	[src] - [dst]

Table 4. Two operand instructions.

These instructions include arithmetic instructions, ADD and SUB. Logical instructions AND, OR, XOR, and BIT. The arithmetic and logical instructions are basically the same as the ones in PDP-11.

One sailent feature of the machine is its extended MOVE and COMPARE instructions. The MOVE instructions can move a word, byte, shape, or a vowel with complete freedom in specifying the source and destination. MOV and MOVB are similar to the ones in PDP-11. MOVS will move the shape code of the character (bits 5 and 6) shown in Figure 3 from source to destination without affecting the rest of the bits in the destination. Source is not affected. Similarly, MOVV will move the vowel code (bits 0 through 4). These two instructions will work properly only if the full character code representation is used.

The COMPARE instructions may be used to compare word, byte, shape, or vowel. Source and destination are not affected. Condition codes are set or cleared according to the result of comparision.

4.5.2 *One-operand instructions*

The format of the one-operand instructions is shown in Figure 8. The operand in the destination field can be specified using any addressing mode. Instructions are listed in Table 5. All of these instructions are similar to the ones in the PDP-11 minicomputer.

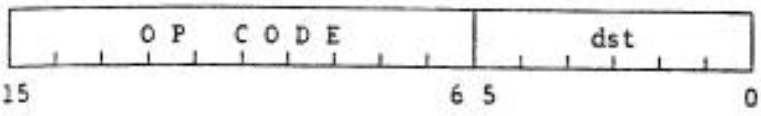

Figure 8. One operand instruction format.

Mnemonic	Meaning	OP code	operation performed
COM	complement	0011 0001	dst ← $\overline{[dst]}$
INC	increment	0011 1000	dst ← [dst] + 1
DEC	decrement	0011 1001	dst ← [dst] - 1
ADC	add carry	0011 1010	dst ← [dst] + [C]
SBC	subtract carry	0011 1011	dst ← [dst] - [C]
LRS	logical right shift	0011 0100	
LLS	logical left shift	0011 0101	
ARS	arithmetic right shift	0011 0110	
ALS	arithmetic left shift	0011 0111	
SWAB	swap bytes	0011 0010	$dst_{15-8} \leftarrow [dst_{7-0}]$ $dst_{7-0} \leftarrow [dst_{15-8}]$

Table 5. One operand instructions.

4.5.3 *Arabic shape and vowel instructions*

The format for this type of instructions is shown in Figure 9. The operand in the destination field can be any general purpose register. These instructions are listed in Table 6.

The put shape instruction (PUTS) will load bits 4 and 5 of the immediate field (IMM) into bits 4 and 5 of the destination register

The put vowel instruction (PUTV) will load the entire immediate operand, that is bits 4 through 8, into the crossponding bit positions of the destination register. These two instructions should be used only when the full character representation is in use.

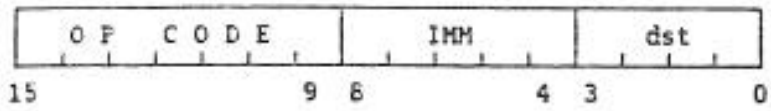

Figure 9. Arabic shape and vowels instruction format.

Mnemonic	Meaning	OP code	operation performed
PUTS	put shape	0011 110	puts a shape code into a register
PUTV	put vowel	0011 111	puts a vowels code into a register

Table 6. Arabic shape and vowels instructions.

4.5.4 *Branch instructions*

Table 7 lists a subset of the branch instructions found in PDP-11. Instruction format is shown in Figure 10.

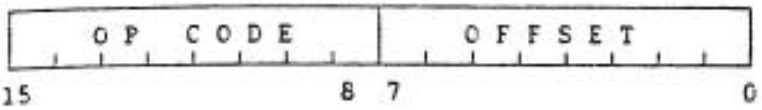

Figure 10. Branch instruction format.

Mnemonic	Meaning	OP code	Branch con.
BR	branch	0000 1000	none
BNE	branch if ≠ 0	0000 0000	Z = 0
BEQ	branch if = 0	0000 0001	Z = 1
BPL	branch if plus	0000 0010	N = 0
BMI	branch if minus	0000 0011	N = 1
BVC	branch if overflow clear	0000 0100	V = 0
BVS	branch if overflow set	0000 0101	V = 1
BCC	branch if carry clear	0000 0110	C = 0
BCS	branch if carry set	0000 0111	C = 1

Table 7. Branch instructions.

4.5.5 *Jump and Subroutine instructions*

Table 8 lists the jump and subroutine instructions they are similar to the ones in PDP-11, their instruction format is included in the OP Code field of Table 8; where DD means a 6-bit destination operand and R means a 4-bit code of a general purpose register.

Mnemonic	Meaning	OP code	operation
JMP	jump	0011 0011 00DD	PC ← dst
JSR	jump to subroutine	0011 0011 01DD	temp← dst SP ← [SP] - 2 [SP] ← [reg] reg ← [PC] PC ← [temp]
RTS	return from subroutine	0011 0011 1000R	PC ← [reg] reg ← [[SP]] SP ← [SP] + 2

Table 8. Jump and Subroutine instructions.

4.5.6 *Condition code and miscellaneous instructions*

Table 9 lists condition code and other operate instructions found in the PDP-11 minicomputer, their instruction format is shown in Figure 11.

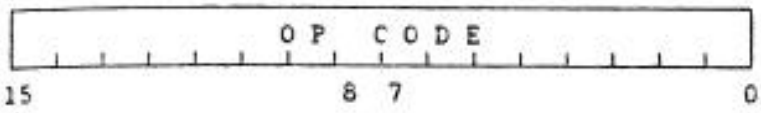

Figure 11. Condition code and miscellaneous instruction format.

Mnemonic	Meaning	OP code	operation
CLC	clear C	00110000 000 10000	C ← 0
CLV	clear V	00110000 000 01000	V ← 0
CLZ	clear Z	00110000 000 00100	Z ← 0
CLN	clear N	00110000 000 00010	N ← 0
SEC	set C	00110000 001 10000	C ← 1
SEV	set V	00110000 001 01000	V ← 1
SEZ	set Z	00110000 001 00100	Z ← 1
SEN	set N	00110000 001 00010	N ← 1
SCC	set all conditions	00110000 011 00000	
CCC	clear all conditions	00110000 010 00000	
HALT	halt	00110000 1100 0000	stops the processor
RESET	reset	00110000 1010 0000	all devices are reset
RTI	return from interrupt	00110000 1001 0000	PC ← [[SP]] SP ← [SP] + 2 PSW ← [[SP]] SP ← [SP] + 2

Table 9. Condition code and miscellaneous instructions.

4.5.7 *Input/Output instructions*

There are 6 I/O instructions listed in Table 10. Their instruction format is shown in Figure 12. The machine will be able to communicate with I/O devices in two ways, by word or by byte.

Output word instruction (OWn) will output to the device specified by the device field (Dev) the contents of the register specified by the register field (Reg).

In case of an output byte instruction (OBn) the most significant byte of the register will be moved to the device specified by the device field.

Input instructions IWn and IBn are similar to the output instructions. ISn is for reading the status of the device and OCn is for giving an output command to the device.

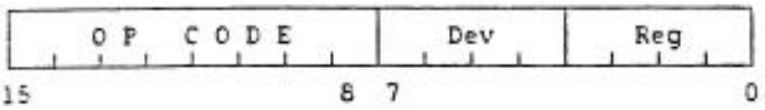

Figure 12. Input/Output instruction format.

Mnemonic	Meaning	OP code	operation
OWn	output word	0000 1010	Output a register to device n
OBn	output byte	0000 1011	Output a byte to device n
IWn	input word	0000 1100	Input a word to a register
IBn	input byte	0000 1101	Input a byte to a register
ISn	input status	0000 1110	Input status of device n
OCn	output command	0000 1111	Output command to device n

Table 10 . Input/Output instructions.

5. ADVANTEGES OF INCLUDING SHAPES AND VOWELS IN THE CHARACTER REPRESENTATION

In this section we will use illustrative examples to elaborate the advantages of storing shape and vowel codes along with the character. One advantage of storing the shape code along with the character is that once we determine the shape of the character at the input, then we do not have to determine it at the output. This will speed up the output and reduce the complexity and cost of output devices. The speed advantage will be more substantial when several outputs of the same text are to be printed.

Another advantage of storing the shape of the character, is that using the shape code in matching Arabic strings will speed up matching process. For example suppose we have an Arabic word (A) and another word (B) as shown below. By the usual matching algorithm the match will stop when the (blank) in (A) is not equal to the the (ta ت) in (B). So 4 characters need to be compared.

A. طس ب

B. ضر بتم

By using the shape code in matching Arabic characters the match will stop where the (Ba ب) in (A) has a different shape than the (Ba بـ) in (B). So, only 3 characters need to be matched. A speed up of 25% in this case. The speed up will be less substantial for longer words, and an average speed up of 15% may be expected.

The third advantage of including the shape code is that it will allow the user to output Arabic characters with a particular shape regardless of its position in the string. For example, compare the Arabic string (A) and (B) given below

(A) اشترى محمد آبام كمبيوتر

(B) اشترى محمد آبم كمبيوتر

Some users will prefer to write string (A) since the abbreviation IBM written as 'آبم' is more clear than 'آ.م'. However, the present day output device will always output string B, or would force the user to insert unnecessary characters, such as blanks, between 'آ', 'ب' and 'م'.

We conclude that storing the shape code will results in the following.

- Faster interaction with I/O devices.
- Reduce the complexity of I/O devices.
- Reduce the cost of I/O devices.
- Speed up matching Arabic strings.
- Give the user more freedom in selecting character shape.

The advantages of using and storing vowels code are as follows:

- Make it possible to use vowels where it is needed.
- Make it possible to use many vowels combination and special signs.
- Reduce the standard (7 bits) code for vowels to 5 bits.
- It will be easier to mix vowelized and unvowelized text.

6. DESIGN AND IMPLEMENTATION TOOLS

Computer aided design tools are necessary to successfully complete the project of this nature in a reasonable time. At our disposal we have an UAHPL (universal AHPL) based automation system. AHPL(A Hardware Programming Language) and its derivative UAHPL have been used at other places to design, model, and test digital systems. The system design of the computer is expressed in UAHPL and processed by the automation system. Simulation runs with benchmark test programs can be made without committing the design to the hardware. Once satisfied with the simulated model the user may ask the automation system to generate the complete design in terms of memory elements, logic gates, buses and their interconnection.

The standard SSI(Small Scale Integration) and MSI (medium Scale Integration) components will be used to implement the machine. It may be possible, in the future, to implement the machine as a VLSI (Very Large Scale Integration) device.

7. CONCLUSION

The arcitectural features and instruction set of a computer especially designed to support Arabic text processing was presented. The approach was compared with the usual approach of using standard CPUs with especially designed I/O devices for processing Arabic text. The advantages of the especially designed CPU were discussed in the previous section. It may be concluded that the system based on the proposed CPU will have improved performance and throughput, lower overall cost, and will provide more flexibility in the handling of Arabic text.

8. ACKNOWLEDGMENT

The authors acknowledge the support of UPM in presenting this paper.

REFRENCES

[1] F. J. Hill and G. R. Patterson, "Digital Systems Hardware Organization and Design, willy, New York, 1978, Second Edition.

[2] V. C. Hamacher, Z. G. Vranesic, S. G. Zaky, "Computer Organization", McGraw-Hill,Inc., New York, 1978.

[3] M. R. Barbacci and D. P. Siewiorek, "The Design and Analysis of instruction Set processors", McGraw-Hill,Inc., New York, 1982.

[4] S. S. Hyder, "A System for Generating Urdu/Farsi/Arabic Script", Proceedings of IFIP congress, 1971.

[5] Saudi Arabian Standards Organization, "Arab standard specifications", proceeding of the international symposium for standardization of codes character sets and keyboards for Arabic language in computers, 1-4 June 1980, Riyadh.

[6] PDP-11 Architecture Hanbook, Digital Equipment Corp., 1983.