Cambridge International **A Level**

Cambridge International Examinations
Cambridge International Advanced Level

CANDIDATE NAME

CENTRE NUMBER | | | | | |

CANDIDATE NUMBER | | | | |

6629823919

COMPUTING **9691/32**

Paper 3 **October/November 2014**

2 hours

Candidates answer on the Question Paper.

No additional materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name on all the work you hand in.
Write in dark blue or black pen.
You may use a soft pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.
No marks will be awarded for using brand names for software packages or hardware.

At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

This document consists of **16** printed pages and **4** blank pages.

CAMBRIDGE International Examinations

1 **(a)** Convert the following infix form expressions into reverse Polish notation.

(i) `(x - y) / 4`

.....[1]

(ii) `3 * (2 + x / 7)`

.....[2]

(b) Convert the following reverse Polish notation expressions into infix form.

(i) `4 a b + c + d + e + *`

.....[1]

(ii) `y 2 ^ z 3 ^ + 5 /`

Note: the caret (^) symbol represents “to the power of”.

.....[2]

(c) **(i)** Describe the operation of a stack.

.....

.....[1]

An expression in reverse Polish notation can be evaluated on a computer system using a stack.

(ii) Give **one** further use of a stack data structure by a computer system.

.....

.....[1]

(d) **(i)** Describe the operation of a queue.

.....

.....[1]

(ii) Give **one** use of a queue data structure by a computer system.

.....

.....

.....[1]

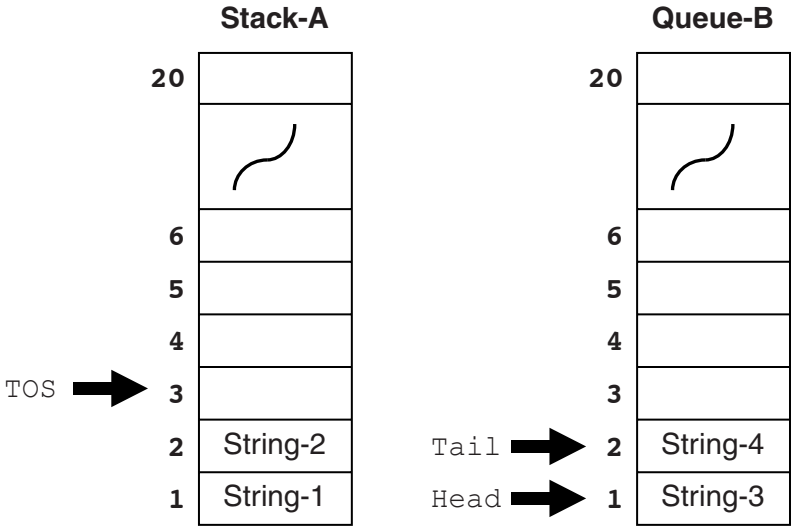
BLANK PAGE

Question 1 continues on page 4.

(e) A stack (Stack-A) and a queue (Queue-B) are to be implemented as follows:
Stack-A has a single pointer `TOS` which shows the index position available for the next new item to be added ('pushed') to the stack.

Queue-B is controlled by two pointers `Head` and `Tail`.

The diagram shows Stack-A and Queue-B after two items have been added to each.



Stack-A and Queue-B are both initially empty.

The sequence of six items of string data:

GOAT CAT SHEEP CHIMP COW HORSE

is stored on Stack-A.

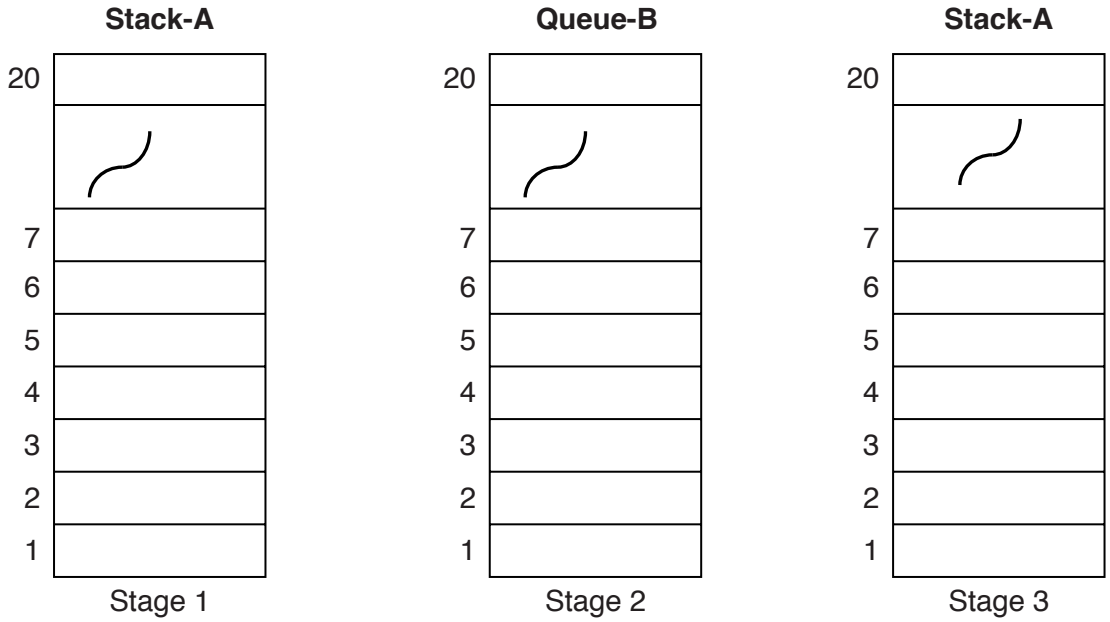
(i) Show on the diagram below, the contents of Stack-A and its pointer at this stage (Stage 1). [2]

The contents of Stack-A are then removed and placed on Queue-B.

(ii) Show on the diagram below, the contents of Queue-B and its pointers at this stage (Stage 2). [3]

Three items are removed from Queue-B and input to Stack-A.

(iii) Show on the diagram below, the contents of Stack-A and its pointer at this stage (Stage 3). [2]



(iv) Three more items are removed from Queue-B and placed on Stack-A.

Comment on the final contents of Stack-A.

.....

.....[1]

2 Modern operating systems use paging for memory management.

(a) An invoicing program is being executed. Some of the pages are stored in main memory. The user selects a print operation. This operation requires a program instruction that is not contained in any of the loaded pages.

Explain how paging deals with this situation.

.....

.....

.....

.....

.....

.....[3]

(b) For a computer system using multi-programming, the low-level scheduler will decide what process will get next use of the processor.

(i) One algorithm could be "give the processor to the process which was given the highest priority".

Describe **two** other strategies which could be used by the low-level scheduler.

1 .....

.....

2 .....

.....[2]

(ii) The computer system runs processes which can be described as either:

- 'processor-bound', or
- 'input/output bound'

Explain each term and give an example.

Processor-bound .....

.....

Example .....

Input/output bound .....

.....

Example .....[4]

(c) A computer system uses interrupts to deal with signals received from a peripheral device such as a printer. The processor is processing Task X when an interrupt is received from Printer P.

The following statements describe how the interrupt is handled:

Label	Step
A	Restore the Program Counter
B	Load and run the appropriate Interrupt Service Routine (ISR)
C	Save the contents of the program counter on the stack
D	Restore all other registers
E	Identify the source and type of the interrupt signal
F	Save contents of all other registers
G	Continue execution of Task X

Complete the sequence of steps using the letters. The final step, G, has been placed for you.

G

[4]

3 A country has a number of cross-country running clubs. Each club organises races which attract runners from other clubs. A database is to be created storing data about races and runners.

The clubs have agreed to stage one race only on any date.

A number of attempts have been made at the database design.

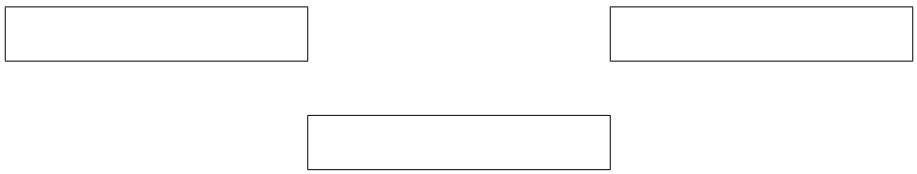
(a) Consider Design 1:

```
Runner(RunnerID, RunnerName, ClubName)

Race(RaceDate, RaceDistance, ClubName)

RaceRunner(RaceDate, RunnerID)
```

(i) Circle the two foreign keys in this database design. [2]

(ii) These three entities form two relationships.

Complete the entity-relationship (E–R) diagram to show them.



[2]

(b) More data is to be stored.

Consider Design 2:

```
Runner(RunnerID, RunnerName, ClubName)
Race(RaceDate, RaceDistance, ClubName,
                                    ClubTown, ClubSecretaryName)
RaceRunner(RaceDate, RunnerID, RunnerName, FinishingPosition)
```

(i) Name the table which is not in Second Normal Form (2NF) and explain why.

Table .....

Explanation .....

.....

.....

Re-design this table.

.....[3]

(ii) Name the table which is not in Third Normal Form (3NF) and explain why.

Table .....

Explanation .....

.....

.....

Re-design this table and add a new table. Both these tables must be fully normalised.

.....

.....

.....

.....[5]

(c) Records have been created for all the runners entered for the race on 26/11/2014.

(i) Write a Data Manipulation Language query to display a list of the IDs of all the runners entered for this race.

.....

.....

.....

.....[3]

(ii) Following the race, the record for runner 8816 must now be updated to show she finished in 2nd place.

Write a Data Manipulation Language command to update this record.

.....

.....

.....

.....[3]

4 The table below gives a subset of the assembly language instructions for a computer with a single general purpose register, the Accumulator (ACC), and an index register (IX).

Instruction			
Opcode (mnemonic)	**Operand**	**Opcode (binary)**	**Explanation**
LDD	<address>	0000 0100	Direct addressing. Load the contents of the given address to ACC
LDV	<number>	0000 0101	Load the given number to ACC
STO	<address>	0001 0000	Store the contents of ACC at the given address
LDI	<address>	0000 0110	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC
LDX	<address>	0000 0111	Indexed addressing. Form the address as <address> + the contents of IX. Copy the contents of this address to ACC
INC	<register>	0000 0011	Add 1 to the contents of the register (ACC or IX)
OUTCH		1000 0001	Output the character corresponding to the ASCII character code in ACC to the monitor
IN		1001 0000	Input a denary number from the keyboard and store in ACC
JMP	<address>	1100 1000	Unconditional jump to the given address
CMP	<number>	1100 1001	Compare the contents of ACC with the given number
JPE	<address>	1110 0111	If the result of the previous compare instruction was a match, jump to the given address

(a) The given table of instructions shows the binary number used for each instruction's opcode.

All instructions in machine code are stored as a 16-bit pattern, with the opcode as the first 8 bits and the operand as the second 8 bits.

(i) What is the maximum number of memory locations which can be directly addressed?

.....[1]

(ii) Consider the instruction:

0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1

Describe what this instruction does.

.....

.....[2]

(iii) Programmers prefer to write machine code instructions in hexadecimal.

Explain why.

.....

.....[1]

(iv) What is the hexadecimal number for the machine code instruction shown in **part (b)(ii)**?

..... [1]

(v) Show the 16-bit machine code for the following instruction:

```
JPE 204
```


[2]

(vi) A programmer makes the statement:

"For this instruction set, some of the instructions do not require an operand"

Circle if this statement is true or false. Explain your choice with reference to the table of instructions given.

True / False

.....

.....

.....

.....[2]

(b) Use the ASCII code table to trace execution of the given program.

ASCII code table (part)					
Character	**Decimal**	**Character**	**Decimal**	**Character**	**Decimal**
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

ACC	Location 450	OUTPUT

```
300  LDI     450
301  CMP      32
302  JPE     308
303  OUTCH
304  LDD     450
305  INC     ACC
306  STO     450
307  JMP     300
308  END

450          500

500           65
501           74
502           65
503           90
504           32
```

[5]

BLANK PAGE

Question 5 begins on page 14.

5 **(a)** An assembler translates assembly language programs. A compiler translates high-level language programs. They each do this for programs written in a particular language and for a particular processor.

Name **two** features which these translators have in common.

1 .....

.....

2 .....

.....[2]

(b) In the early stages of program development, a source program usually has errors. Each statement below (labelled A, B, C, D, E, F, G and H) describes one step in the development of the machine code version of an assembly language program.

Note: One of the steps given is not relevant and will not be used.

Label	Step
A	Generate executable file
B	Make changes to the source code
C	UNTIL No errors reported
D	Run the assembler with the executable code
E	REPEAT
F	Run the assembler with the source code
G	Write the assembly language source program
H	Report all error(s)

Put the steps in the correct sequence to describe the assembly process.

The first two steps, G and E, have been placed for you.

G
E

[5]

(c) A developer writes programs in a high-level language. Both an interpreter and compiler exist for the language.

(i) Describe **two** benefits that the use of an interpreter would offer.

.....

.....

.....

.....[2]

(ii) Describe **one** drawback in the use of an interpreter.

.....

.....[1]

6 **(a)** The sequence of operations below shows the fetch stage of the fetch-execute cycle in register transfer notation.

```
1. MAR ← [PC]
2. PC ← [PC] + 1
3. MDR ← [[MAR]]
4. CIR ← [MDR]
```

Note: `[register]` denotes the contents of the specified register.

Explain what is happening at the fetch stage.

1 .....

.....

.....

2 .....

.....

.....

3 .....

.....

.....

4 .....

.....

.....[4]

(b) The address bus and data bus are used during the fetch-execute cycle.

(i) Name another bus used in a typical microprocessor.

..... [1]

(ii) Name **one** signal carried by this bus.

.....

.....[1]

(c) Consider **two** assembly language instructions which were given in **Question 4**.

Instruction		Explanation
Opcode (mnemonic)	**Operand**	
`INC`	`<register>`	Add 1 to the contents of the register (ACC or IX)
`LDD`	`<address>`	Direct addressing. Load the contents of the given address to ACC

Consider the following two cases.

Case 1:
Following step 4 of the fetch stage, the instruction is decoded. The instruction is then executed without further use of the address bus.

Case 2:
Following step 4 of the fetch stage, the instruction is decoded. Once decoded, the address bus must be used again before the execution of the instruction can be completed.

For each instruction below, circle either Case 1 or Case 2 and explain your choice.

(i) `LDD 78`

Case 1 / Case 2

Explanation: .....

.....

.....[2]

(ii) `INC ACC`

Case 1 / Case 2

Explanation: .....

.....

.....[2]

7 The function `Replace` is documented as follows:

```
FUNCTION Replace(ThisString : STRING, FindString : STRING,
              ReplaceString : STRING, [MatchType : CHAR]) RETURNS STRING
```

Square brackets used in the function header indicate the parameter is optional.

The function searches for `FindString` within `ThisString`. When a match is found, all occurrences of `FindString` within `ThisString` are replaced with `ReplaceString`.

`MatchType` indicates the type of match looked for:

- `'E'`: indicates the strings must match exactly
- `'C'`: indicates a match is reported irrespective of the case of either string

The function returns `ThisString`.

Any function call which is not properly formed returns an error.

What is returned by the following function calls?

(a) `Replace("Mary Adams", "Adams", "Kelly", 'E')`

.....[1]

(b) `Replace("10110011", "011", "X", 'E')`

.....[1]

(c) `Replace("Dr Ajaz Drew", "Dr", "", 'E')`

.....[1]

(d) `Replace("database design", "sign")`

.....[1]

(e) `Replace("white box strategy", "STRATEGY", "TESTING", 'C')`

..... (1)

(f) High-level programming languages have two types of function. These are **built-in** and **user-defined**.

Explain the difference between them. You may give an example from your practical experience for a built-in function.

.....

.....

.....

.....[2]

BLANK PAGE

BLANK PAGE