GCE 2004
June Series

ASSESSMENT and QUALIFICATIONS ALLIANCE

Mark Scheme

Computing
Unit CPT4

Mark schemes are prepared by the Principal Examiner and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation meeting attended by all examiners and is the scheme which was used by them in this examination. The standardisation meeting ensures that the mark scheme covers the candidates' responses to questions and that every examiner understands and applies it in the same correct way. As preparation for the standardisation meeting each examiner analyses a number of candidates' scripts: alternative answers not already covered by the mark scheme are discussed at the meeting and legislated for. If, after this meeting, examiners encounter unusual answers which have not been discussed at the meeting they are required to refer these to the Principal Examiner.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of candidates' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this Mark Scheme are available from:

Publications Department, Aldon House, 39, Heald Grove, Rusholme, Manchester, M14 4NA
Tel: 0161 953 1170

or

download from the AQA website: www.aqa.org.uk

Set and published by the Assessment and Qualifications Alliance.

The Assessment and Qualifications Alliance (AQA) is a company limited by guarantee registered in England and Wales 3644723 and a registered Charity number 1073334. Registered address AQA, Devas Street, Manchester. M15 6EX. *Dr Michael Cresswell Director General*

Computing: Unit CPT4

The following notation is used in the mark scheme

- **; - means a single mark;**
- **/ - means alternative response;**
- **A – means acceptable creditworthy answer;**
- **R – means reject answer as not creditworthy;**
- **I – means ignore.**

1. (a) (i) city_in (london, uk); **1**

(ii) visited (strasbourg); **1**

(b) france; uk; **2**

(c) visited (City); AND city_in; (City, Country).;
or city_in; (City,Country); AND visited(City); **3**

Total 7

accept any variable name instead of City, but must be same in both places penalise once for wrong case (atoms, variables and predicates only)
A and
then *.....t.o. of one mark*
I *fullstop and any spurious punctuation*

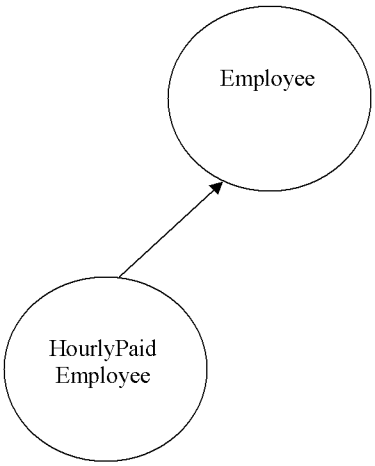
2. (a) from -2^{15}/-32,768; to 2^{15}-1/32767; **R** *binary;* **R** *one number unqualified* **2**

(b) 2^{10}-1 / 1023; **A** 1111111111_2; **A** &3FF; **A** $3FF_{16}$;

(c) Step 1: (Content of) PC copied into MAR // [PC] → MAR; *accept {} for []*
Step 2: (Content of) PC incremented (by 1) // [PC] incremented (by 1);
Step 3: content of Memory location addressed by MAR; loaded into MDR;//
MAR addresses a memory location; whose content is loaded into MDR;
Step 4: Content of MDR copied into CIR // [MDR] → CIR;
Step 5: Content of CIR decoded // [CIR] decoded // opcode/instruction/data decoded;
Step 6: instruction executed; **max 6**
step 2 could be after step 3,4 or step 5 but otherwise
order of steps must be as above, step(s) may be missed out
steps do not have to be one per line
P1 *of it is difficult to pick out the steps from prose.* **9**

A loaded/moved/stored for copied
A MBR for MDR
A IR for CIR
A SCR for PC

3. (a)



2

1 mark if correct hierarchy (including rectangles or round/oval shapes) in an inheritance diagram;
A *no shapes this year only*
1 mark for arrow in correct direction;

(b) different objects can respond to the same message in different ways;
Giving an action one name that is shared up and down an object hierarchy
with each object in the hierarchy implementing the action in a way appropriate to itself;
same name used in the class hierarchy for a method/action but each class implements this method differently; **max 1**

A *an example in context such as 'add' which behaves differently if used with strings than if used with numbers*

(c) THourlyPaidEmployee = Class (Employee)
(Public)
procedure CalculatePay (override)
procedure GetNumberOfHoursWorkedInMonth
Private
hourlyrate/hourlypay/HourlyPayRate: Currency
NumberOfHoursWorkedInMonth : Integar/Real/Float
End
OR

public class/subclas THourlyPaidEmployee extends/inherits TEmployee; **1**
{
public void calculatePay; **1**
public void getNumberOfHoursWorkedInMonth; **1**
private; float hourlyPayRate; **1 mark for private, 1 mark for var name**
private int numberOfHoursWorkedInMonth; **1**

Sub total 6

Total 9

}
Accept "Object" instead of "Class"
Accept Public implied
Lose one mark if properties from parent class included
R *any diagrams*

4. (a)

18: 0000 0000 0001 0010;

-6 1111 1111 1111 1010;

---

12 0000 0000 0000 1100; *if previous binary patterns correct* **4**

I mark for showing 16 bits throughout

(b) (i)

Symbolic Address	Hexadecimal Representation	Binary Representation	Decimal Value
Num2	A802	**1010 1000 0000 0010;**	**-2.75;;;**

if answer wrong give:
moving e places to right / exponent processing 2^e or equivalent: 1 mark
correctly identifying negative number 1 mark **4**
follow through if binary representation wrong

(ii) To maximise precision in a given number of bits // to minimise rounding errors
// to have just one representation of the decimal number // to simplify arithmetic operations; 1 mark
A to maximise accuracy in a given number of bits; **9**

5. (a)

Opcode	Operand(s)	Comment
LDA/ MOV/MOVE	3C5/ 3C5,R0 / R0,3C5;	Load contents of 3C5 into Accumulator *1 mark for correct opcode and operand*
OR;	#0010 0000B;; #&20;; #32;;	Logical OR with immediate address: set bit 5 (6th bit from the right). *1 mark for #/immediate addressing* *1 mark for correct value(accept without & or B)*
STA/ MOV/MOVE	3CB/ R0,3C5 / 3C5, R0;	Store result in memory location 3CB *Direction of MOV must be consistent with MOV statement above* *1 mark for correct opcode and operand*

Must have comments if unusual opcodes **max 4**

(b) it would remain in lower case//no change; **1**

Alternative answer:

(a)

Opcode	Operand(s)	Comment
LDA/ MOV/MOVE	3C5/ 3C5,R0 / R0,3C5;	Load contents of 3C5 into Accumulator *Direction of MOV must be consistent with MOV statement below* *1 mark for correct opcode and operand*
OR;	#0010 0000B;; #&20;; #32;;	Logical EOR/XOR with immediate address: set bit 5 (6th bit from the right). *1 mark for #/immediate addressing* *1 mark for correct value(accept without & or B)*
STA/ MOV/MOVE	3CB/ R0,3C5 / 3C5, R0;	Store result in memory location 3CB *Direction of MOV must be consistent with MOV statement above* *1 mark for correct opcode and operand*

Must have comments if unusual opcodes **max 4**

(b) it would change to upper case; **1 mark** **5**

part (b) must follow from consistent use of OR but
follow through if ADD is used instead of OR: ASCII code would be for a different character;

6 (a) code which can be loaded into any area in memory
/into a different area in memory each time it is run; **R** save instead of loaded **1**

(b) effective address = content of base register + offset/displacement/number;;;
in multi-programming o.s. all address references use base register addressing;
when a program is loaded into main memory;
the base register is loaded with address of base of memory block containing program;
so programs may be relocated/work anywhere in main memory;
base register addressing produces relocateable code;
every address is relative to the base register; **max 5** **6**

7 (a)

interactive	**batch:**	
user and computer in two way communication;	when processing begins, it continues from beginning to end without user interaction/intervention;	**max 1**
user communicates directly with program as it is running;	Program run in background with no interaction with the user;	**max 1**
processing carried out as users enter the data so results are available immediately;	Processing delayed until all data have been entered;	**max 1**

I purpose **max 2**

(b) job ID; priority; user name; job delimiters; job completion time;
approx execution time; max length of time job can run for; start time of job;
main memory required; devices/hardware required (eg printer);
compiler/assembler/software required; name of file/program to be executed;
what to do on non-successful completion of job;
data file required; output destination; **max 3**

(c) job requires i/o transfer; **A** process instead of job **R** program instead of job
job requires non-shareable resource which is already allocated;
job involves communicating with another process and expected message not arrived yet;
(some) error condition; **R** fatal error
operator intervenes to temporarily halt process; **max 2**

(d) circular queue//queue//linked list//priority queue//linked list of records;
R array *on its own* **1**

(e) the mix of processor and i/o bound jobs currently active;
the utilisation of hardware resources such as printer/memory;
the utilisation of processor // active list gets too short // room on active list //active list empty;
proximity of a deadline for a batch job;
time of day as interactive jobs tend to cease during night;
(batch) job in active list terminates therefore allowing another (batch) job to run;
priority;
time waiting in inactive list; **Max 2**

Total 10

8 (a) a procedure/routine which calls itself//is defined in terms of itself; **R** re-entrant; **1**

A function instead of procedure **R** program
iteration Talked Out (no mark)

(b) (i)

E	L	H	M	List[M]	Printed Output
6502	1	11 ;	6	5789 ;	
6502	7	11 ;	9	8407 ;	
6502	7	8 ;	7	6502 ;	
					True;

Accept True *in row 3*
Marks in each row for all three/two parts correct
Accept empty cell to mean: same as in previous row.
Stop marking when logic goes wrong **7**

(ii) binary search;;
search;
R any other type of search **2**

Total 10

END OF CPT4 MARK SCHEME