Concordia University
Department of computer science

Midterm test

Professor : *Kerly Titus*
Course : *System software (COMP229/1-CA)*
Date : *Monday, Thursday July 26, 2001*
Time : *17 :30 – 18:45*
Special instructions : - *Answer all questions on the booklet*
- *No documentation allowed*

Question 1. Assemblers [21 marks]

a) In the SIC/XE assembler, only the memory locations of constants declared with the BYTE or WORD directives are initialized when the object program is loaded into the main memory. Consequently, the variables declared with the RESB or RESW directives are not initialized and their memory locations contain unknown values.

- Explain how the assembler for SIC/XE could be modified so that the data that are declared with RESB are initialized to spaces (i.e. " ") and the data declared with RESW are initialized to 0. Specify clearly the changes that apply to the assembly language source code and to the object program, and the pass that each change occurs.

- Is it really worth to perform this initialisation, or could it be preferable to let the programmer handle this task? Explain.

b) Explain how a load-and-go assembler processes forward references. Can a load-and-go assembler processes source programs that contain control sections? Explain why or why not.

c) Literal operands can be allocated memory either at the end of the program or within the program where an LTORG directive is specified.

- Explain the advantage that is obtained when the literals are generated within the program.
- Explain how the literals that are generated by the LTORG directive can be allocated contiguous memory locations in a common area.

Question 2. Loaders and linkers [17 marks]

a) Control sections use a REFER directive to indicate to the assembler the symbols that are externally defined.

- What would happen at assembly time if the programmer omits to specify this REFER record in the source code?

- Is that REFER record used later by the linker? Explain.

b) Explain what type of loader (absolute or relative) uses modification records and what type uses relocation bits to perform relocation. Explain the problem that may occur if a loader applies the other relocation technique.

c) Explain how a dynamic link library (DLL) allows the computer system to save memory and disk space. Explain a situation for which a dynamic library would use more space than a static library.

Question 3. Introduction to operating systems [12 marks]

a) Explain the difference between low-level and high-level abstraction. Can an operating system implement both abstraction levels? If yes, explain a situation that requires a low-level abstraction and a high-level abstraction.

b) Explain what are space-multiplexed and time-multiplexed resource sharing. Give an example for each.

c) Explain why the Linux operating system is classified in the multiprogramming category and not in the multitasking category.

Concordia University
Department of computer science

Midterm test - solution

Student name : _____ **Student ID :** _____
Professor : *Kerly Titus*
Course : *System software (COMP229/2-S)*
Date : *Friday, October 20 2000*
Time : *8:45-10:00*
Special instructions : - *Answer all questions on the questionnaire*
- *No documentation allowed*

Question 1. Assemblers [25 marks]

a) Explain the main reason why a one-pass assembler requires the programmer to declare any variable or constant before the executable statements. Can this requirement allow the one-pass assembler to process any program without restrictions? Explain. [5 marks]

- *A one-pass assembler resolves the symbolic operands and converts the symbolic instructions in one scan of a source program. In order to do so, it is required to know the addresses of the symbolic operands prior to instruction conversion. Thus, the declarative statements must always precede the imperative statements.*

- *By placing the declarations before the instructions is not sufficient to prevent the occurrence of forward references. Most programs make use of forward jumps in instructions and this situation is almost unavoidable.*

b) Explain how literals are processed by the assembler. What is an advantage to use literal operands? [7 marks]

- *Literals are inserted in a literal table when they are processed by the assembler. Memory is reserved for the literals when the END or LTORG directive is processed.*

- *An advantage of the use of literals is that the programmer does not have to explicitly define the constants.*

c) Explain the main usage of the EQU directive. Can all symbols that are declared with an EQU directive be resolved in one pass? Explain. [5 marks]

- *The EQU directive is used to define aliases of program symbols such as constants, statements, keywords, etc.*

- *An EQU directive can define an alias for an expression having forward terms and requiring multiple passes.*

d) Explain the main difference that exists between program blocks and control sections. Can both be used in a same program? [8 marks]

- *A program block represents a logical organization of program statements where code and data are separated within the same object program.*

- *A control section represents a grouping of independent program modules in separate files. The modules are usually part of a same application program and require to be combined by a linker prior to execution.*

- *Both can be used in a same program because a control section can store a complete program, which may be divided into program blocks.*

Question 2. Loaders and linkers [17 marks]

a) Explain the main difference between a linking-loader and a linkage editor. Specify a situation where it would be advantageous to use a linking-loader, and another for a linkage editor. [6 marks]

- *A linking-loader performs both linking, loading, and execution in one step.*

- *A linkage editor only links control sections without loading and executing the application program. It generates an executable file that needs to be processed by a loader for execution at any given moment.*

- *A linking-loader would be useful in a system development phase where an application program is often assembled, linked, and executed.*
  A linkage editor would be useful in situations where it is not necessary to execute the application program immediately. That situation could be in the final phase of a system development prior to delivery to customer.

b) Explain how dynamically loaded routines are processed when called by a user program at run time. What is an advantage to use dynamic linking instead of static linking? [6 marks]

- - *User program makes a system call to load a dynamic routine.*
  - *Dynamic loader loads the routine in memory and executes it.*

- *OS returns execution control to the user program and may unload the routine.*

c) Explain the reason why a bootstrap loader must be absolute whereas a program loader can be relative or absolute. When a program loader is required to be relative? [5 marks]

- *A bootstrap loader is only dedicated to load the code of the OS at start up. During start up of the system, there is no multitasking or multiprogramming so memory is not shared. Therefore, the OS is always loaded at the same location.*

- *A program loader can be relative only if it will execute in a multitasking or multiprogramming environment where memory is shared by multiple processes. Otherwise, a program loader can be absolute but being relative will not cause any problem.*

Question 3. Operating systems [8 marks]

a) Explain what are resource sharing and resource abstraction. [4 marks]

- *Resource sharing is an objective of an OS that allows it to share the system resources among competing processes.*

- *Resource abstraction is another objective of an OS and it provides a high-level interface to processes and users to access the system resources.*

b) Explain the main difference that exists between a batch and a time-sharing system. Can an operating system implements both? Explain. [4 marks].

- *In a batch system, all commands to the OS are processed with no user interaction through a console. In such environment, there is no time limit imposed to processes during execution. The only time a process may interrupt and relinquish the CPU to another process is during an I/O operation.*

- *In a time-sharing system, the user sends the commands to the OS interactively through the console. In addition, each process is usually assigned a fixed time slice to use the CPU and is often interrupted after this deadline.*

- *Modern OS include both batch and time-sharing capabilities. Batch functionality is used to allow a user to automate the processing of related group of commands. Time-sharing is useful if the user needs to execute many processes simultaneously.*

 24

Concordia University
Department of computer science

Midterm test - solution

Student name : _____ **Student ID :**_____
Professor : *Kerly Titus*
Course : *System software (COMP229/2-MM)*
Date : *Tuesday, October 17 2000*
Time : *17 :45 – 19:00*
Special instructions : - *Answer all questions on the questionnaire*
- *No documentation allowed*

Question 1. Assemblers [28 marks]

a) In the SIC or SIC/XE assembler, the variables that are declared with the directives RESB and RESW are not initialized. In that case, if the programmer does not explicitly provides initial values then these variables will contain unexpected values. [8 marks]

- Explain how the SIC or SIC/XE assembler could be modified so that the variables are initialized with default values (blank spaces for bytes and zeros for words).

The only way to initialize memory locations is to set the initial values in a text record of the object program. In that case, the assembler must write text record entries for RESB and RESW directives, just like it does for BYTE and WORD directives.

- Explain a programming situation (e.g. a statement) that would absolutely necessitate the initialization of the variables.

A counting or incrementing variable (e.g. x = x + 1) must be initialized to a start value by the programmer or to a default value by the assembler or compiler.

b) Literal and immediate values are both data that are introduced as symbolic operand in an instruction. [7 marks]

- Explain the main difference that exists between literal and immediate values.

A literal is assembled similarly to a constant and is reserved a memory location by the assembler. An immediate value is coded directly in the operand field of the machine instruction and does not require any memory representation.

- Explain the main advantage that is obtained to use an immediate value instead of a literal value.

  An immediate value is processed faster because there is no operand fetch required to obtain the value. The value is actually fetched as part of the instruction that contains the immediate operand.

c) Explain the main usage of the ORG directive in the SIC or SIC/XE assembler. What problem may occur when this directive is used? [5 marks]

- *The ORG directive is used to change the current value of the location counter (LC) during the assembling phase of a SIC/XE program.*
- *Using the ORG directive may cause a problem in subsequent memory allocations for instructions and data.*

d) Explain the main difference that exists between program blocks and control sections. Can both be used in a same program? Explain. [8 marks]

- *A program block represents a logical organization of program statements where code and data are separated within the same object program.*
- *A control section represents a grouping of independent program modules in separate files. The modules are usually part of a same application program and require to be combined by a linker prior to execution.*
- *Both can be used in a same program because a control section can store a complete program, which may be divided into program blocks.*

Question 2. Loaders and linkers [15 marks]

a) Explain the main difference between a linking-loader and a linkage editor. Specify a situation where it would be advantageous to use a linking-loader, and another for a linkage editor. [6 marks]

- *A linking-loader performs both linking, loading, and execution in one step.*
- *A linkage editor only links control sections without loading and executing the application program. It generates an executable file that needs to be processed by a loader for execution at any given moment.*
- *A linking-loader would be useful in a system development phase where an application program is often assembled, linked, and executed.*
  A linkage editor would be useful in situations where it is not necessary to execute the application program immediately. That situation could be in the final phase of a system development prior to delivery to customer.

b) Explain how the loader processes the external references that are made between control sections during the linking phase. [4 marks]

In pass one, the linker resolves all external references by placing them in an external symbol table (SYMTAB). This table contains the base address of control sections, and the address of their relative external references.
When the linker processes the modification records, it adds the base address of a control section to the relative address of an external reference.

c) Explain the reason why a bootstrap loader must always be absolute whereas a program loader can be relative or absolute. When a program loader is required to be relative? [5 marks]

- *A bootstrap loader is only dedicated to load the code of the OS at start up. During start up of the system, there is no multitasking or multiprogramming so memory is not shared. Therefore, the OS is always loaded at the same location.*

- *A program loader can be relative only if it will execute in a multitasking or multiprogramming environment where memory is shared by multiple processes. Otherwise, a program loader can be absolute but being relative will not cause any problem.*

Question 3. Operating systems [7 marks]

a) Explain what are resource sharing and resource abstraction. [4 marks]

- *Resource sharing is an objective of an OS that allows it to share the system resources among competing processes.*

- *Resource abstraction is another objective of an OS and it provides a high-level interface to processes and users to access the system resources.*

b) Explain why Linux is a multiprogrammed operating system and Windows 98 is Multitasking operating system. [3 marks]

- *Linux is multiprogrammed because it can handle many user connections (sessions) and it can also execute many programs concurrently.*
- *Windows 98 is multitasking because it can only execute concurrent processes of a same user.*

Concordia University
Department of computer science

Midterm test

Professor : *Kerly Titus*
Course : *System software (COMP229/1-CA)*
Date : *Thursday, July 27 2000*
Time : *17 :30 – 18 :50*
Special instructions : - *Return questionnaire*
- *Answer all questions in the examination booklet*
- *No documentation allowed*

Question 1. *Assemblers [23/50]*

a) Explain how a load-and-go one-pass assembler handles forward references. [8]

- ***The operand is entered in the symbol table with an undefined address value.***
- ***The address of the operand field is added to a list of forward references which depend on the address of this operand.***
- ***When the operand is declared, the list of operands is processed and resolved.***

b) Explain the usage of program blocks and control sections. Can both be used in a same program? Explain. [9]

- ***A program block is used to reorganize a source program into logical segments. The program statements are thus grouped in data (small or large) and instruction blocks which are handled separately by the loader. This reorganization allows to concentrate data and instructions in contiguous sections.***

- ***A control section represents a physical division of a source program into independent segments. The property of independence means that a segment may have its own symbol definitions and instructions, and may be assembled separately.***
  Control sections can be stored in libraries and be accessible by applications that request their services. Thus, it is not necessary that each application rewrites and defines the code for the control section.

- ***A control section has a similar structure to a source program, it has also data and instructions which can be grouped into program blocks.***

c) Explain the main difference that exists between a constant declaration (BYTE or WORD directive) and an EQU directive. [6]

- *A constant requires a memory location that is initialized with the constant value.*
- *The operand of an EQU directive is coded as an immediate value in the operand field of the instruction and does not require memory reservation.*
- *The processing of an EQU symbol is more efficient than a constant because no data fetch cycle is necessary to obtain the operand. However, a constant can be a structured data whereas an EQU symbol can only be a single data.*

Question 2. *Loaders and linkers [17/50]*

a) Explain the reason why the relocation bit approach can not be used with the SIC/XE machine. The relocation bit is a technique that is used by the assembler to provide relocation information for the loader. [6]

- *Each relocation bit applies to an instruction at a relative position within a given text record. As the relocation bit specifies which instruction needs to have its operand reference relocated but not the relative address of the reference, it is assumed that the instruction size is fixed. If the instruction size is not fixed, misalignment will occur and the loader will not point to the appropriate operand reference to relocate.*

b) Explain why it is preferable to use the relocation bit technique with the SIC computer rather than using modification records. [5]

- *The SIC computer uses mainly the direct addressing mode which would require a modification record for almost every instruction. Therefore the size of the object program would be very large and the processing of the assembler would be inefficient.*

c) Explain how a linking-loader handles the define and refer records that are specified in an object program. [6]

- *During pass one, the loader processes each control section separately and assigns an address to each. For each define record, the symbol is entered in the external symbol table with its address value relative to the start of its control section.*
- *For each symbol in a refer record, modification records that have been generated for operands that reference to them are processed during pass two.*

Question 3. *Introduction to operating systems [10/50]*

a) Briefly explain the following terms. [8]

- Resource abstraction.

  - *The process by which the operating system (OS) hides the internal characteristics of a resource to the user. The user interacts with a resource using high-level commands*

that make no reference to any physical characteristics of the resource. The same high-level command may be used to abstract different resources.

- Space-multiplexed resource sharing.
  - ***The resource is sharable and its units are allocated simultaneously to different processes.***
- Time-multiplexed resource sharing.
  - ***The resource is indivisible and is allocated only to one process at any specific instant.***
- Multitasking.
  - **A concept that allows an operating system to execute multiple program instances concurrently.**

b) Briefly explain a major difference that exists between Unix and Windows 98 operating systems. [2]

- ***Unix is a multi-user operating system whereas Windows 98 can only handle a single user at a time.***

27

Concordia university
Department of computer science

Midterm test

Professor : *Kerly Titus.*
Course : *System software (COMP229/2-N).*
Date : *Friday, October 22 1999.*
Time : *8 :45-10 :00.*
Special instructions : *Return questionnaire.*

Question 1. *Assemblers (22/50)*

a) Explain the main steps of a two-pass assembler. (6)

- ***Pass one.***
  - ***The source program is scanned to build a symbol table and to validate the statements. When a label definition is encounterd, this label is assigned the value of the location (LC) counter and entered in the symbol table. The value of the location counter is incremented by the size of an instruction or a symbol (variable or constant).***

- ***Pass two.***
  - ***The intermediate file generated during the first pass is processed and machine codes are generated for the symbolic statements. The symbol table is looked up to determine the numeric addresses of the symbolic references in the converted instructions.***

b) Explain the usage of program blocks and control sections. Can both be used in a same program? Explain. (9)

- ***A program block is used to reorganize a source program into logical segments. The program statements are thus grouped in data (small or large) and instruction blocks which are handled separately by the loader. This reorganization allows to concentrate data and instructions in contiguous sections.***

- ***A control section represents a physical division of a source program into independent segments. The property of independence means that a segment may have its own symbol definitions and instructions, and may be assembled separately.***
  Control sections can be stored in librairies and be accessible by applications that request their services. Thus, it is not necessary that each application rewrites and defines the code for the control section.

- ***Because a control section has a similar structure to a source program, it has also data and instructions which can be grouped into program blocks.***

c) Explain one way that literal operands are processed by an assembler. Explain whether literals are machine-dependent or machine-independent. (7)

- ***The assembler generates literal table entries for the literals which are assigned addresses (using LC) from the end of the object program or at the occurrence point of the LTORG directive.***

- ***Literals are machine-independent.***

Question 2. *Loaders and linkers (13/50)*

a) Suppose that a computer primarily uses direct addressing, but has several different instruction formats. What problems does this create for the relocation bit approach to program relocation? (5)

- ***Each relocation bit applies to an instruction at a relative position within a given text record. As the relocation bit specifies which instruction needs to have its operand reference relocated but not the relative address of the reference, it is assumed that the instruction size is fixed. If the instruction size is not fixed, misalignment will occur and the loader will not point to the appropriate operand reference to relocate.***

b) Consider the following program skeletons. (8)

```
0000    PROGA   START 0
        EXTDEF  X1
        EXTREF  Y2
        ...
0015            LDA     X1
0018            ADD     Y2
0021            STA     Z1
        ...
0060    X1      RESW    1
0063    Z1      RESW    1
                END

0000    PROGB   START 0
        EXTDEF  Y2
        EXTREF  X1
        ...
0015            LDA     X1
0018            ADD     Y2
0021            STA     Z2
```

```
...
0060    Y2          RESW    1
0063    Z2          RESW    1
                    END
```

- Show the external symbol table that is generated when PROGA and PROGB are linked by the linking loader. Assume the load address is 5000_h.

Control section	Symbol name	address	Length
PROGA		5000	0066
	X1	5060	
PROGB		5066	0066
	Y2	51C6	

Question 3. *Operating systems (15/50)*

a) Briefly explain the following terms. (7)

- Resource abstraction.

  The process by which the operating system (OS) hides the internal characteristics of a resource to the user. The user interacts with a resource using high-level commands that make no reference to any physical characteristics of the resource. The same high-level command may be used to abstract different resources.

- Space-multiplexed resource sharing.

  - ***The resource is sharable and its units are allocated simultaneously to different processes.***

- Time-multiplexed resource sharing.

  - ***The resource is indivisible and is allocated only to one process at any specific instant.***

b) Explain the main difference that exists between batch and time-sharing systems in terms of command processing. Is it possible that an operating system provides both batch and time-sharing modes? (4)

- ***In a batch system, the commands are issued using a batch file (or a job queue) whereas a time-sharing system allows the user to interact directly with the computer. Also, in typical batch operating systems a job is scheduled until completion of its task.***
- ***Contemporary operating systems include both batch and interactive command processing. The advantage of the batch file is that it avoids the user to have to type a sequence of commands repeatedly.***

c) When a program is allocated the use of the CPU for execution, does it keep the resource permanently until it terminates? Explain. (4)

- ***In most operating systems each process is allocated a fixed time unit (or quantum) to execute and at the end of this time the CPU is prempted and allocated to another process. The suspended process is blocked on a ready queue and must wait to be later rescheduled.***

27

Concordia university
Department of computer science

Midterm examination - solution

Course : System software (COMP229/1-CA)
Instructor : Kerly Titus
Date : Tuesday, July 27 1999.
Time : 18:30 – 19:45
Attached documents : - SIC & SIC/XE instruction set.
- SIC & SIC/XE machine instruction formats.

Question 1. *(25/60)*

a) Specify and briefly explain 3 architectural features that make SIC/XE more advantageous than SIC for processing programs. *(9)*

- The register set.
  - Faster operations.
  - Shorter instruction formats with register-register operands.
- The memory.
  - Larger memory space.
  - Multiprogramming capabilities.
  - CPU time used more efficiently.
- The addressing modes.
  - Various addressing modes provide more flexibility and efficiency.

b) Explain the principles of a two-pass assembler. *(9)*

- A location counter (LC) keeps track of the current location of an instruction during the assembly scan (pass one). It is initialized to the start address of the program (usually zero), and is incremented by the size of an instruction, a variable or a constant. The size of the variables or constants can be determined from directives such as RESW, WORD, BYTE, or from specifications in the op-code table. The op-code table is also looked-up to validate the mnemonic statements.
- When a symbolic reference is encountered, the assembler uses the location counter to associate the symbol to an address in the symbol table. Once an entry is inserted in the symbol table, the subscript associated with the table is incremented by 1.

```
        END     Sample
```

a) Show the listing file for the program above. (20)

```
0000  Sample  START   0
0000  first   STL     retadr   172014
0003          LDCH    K30      532010
0006  loop    TD      output   E3200C
0009          JEQ     loop     332FFA
000C          WD      output   DF2006
000F          LDL     retadr   0B2005
0012          RSUB             4C0000
0015  output  BYTE    X'05'    05
0016  k30     BYTE    C'0'     30
0017  retadr  RESW    1
              END     Sample
```

b) In the SIC & SIC/XE assembly language programs, the definitions of constants always precede the definitions of variables. For instance, the constants output and k30 are defined before the variable retadr in the above program. *(5)*

- Explain the effect on the object code if the variables would be defined before the constants.

  There are no data loading required for the variables and consequently they have no entry in the object code. If we define the variables prior to the constants, we have to start a new text record because the load addresses in the object program would be misaligned.

Question 3. *(10/60)*

Consider the following layouts for a main program and an external routine. The external routine will be linked with the main program.

```
0000  PROGRAM   START   0
      EXTDEF    P1, P2
      EXTREF    R1
      ...
      ...
002A  LDA       R1
      ...
      ...
00F0  P3        WORD    P1 – P2
      P1        RESW    1
      P2        RESB    20
        ...
      ...
```

```
                ...
0C23                    END     PROGRAM

0000    ROUTINE CSECT
        EXTREF  P1, P2
        EXTDEF  R1
        ...
        ...
001C    LDS     P1
001F    LDT     P2
0022    SUBR    S, T
0024    STT     R1
        ...
        ...
003A    R1      RESW    1
        ...
        ...
        ...
0040            END
```

a) Assuming that the linked program is loaded at memory address 2000h, show the external symbol table generated by the loader for the above sample codes. *(8)*

Control section	*Symbol name*	*Address*	*Length*
PROGRAM		2000	0C24
	P1	20F3	
	P2	20F6	
ROUTINE		2C23	0041
	R1	2C5D	

b) How many modification records would be generated in the object code for the sample codes above. Explain briefly. *(2)*

- A modification record is required for the statements LDA R1, LDS P1, and LDT P2 because their operand references are external. The other statements use register or displacement operand references and do not require relocation.

In addition to the mnemonic and machine language statements, the symbol table may also list the error condition flags, the data type, and the data length.

- After the source code has been scanned totally and the symbol table fully implemented, the assembler generates machine code on the second pass. Hence, when an instruction has a symbolic operand reference, the assembler then looks-up in the symbol table and replaces the symbol with the appropriate value in the machine instruction that it generates. Also, values are generated for constant memory references and some directives that were not assembled during the first pass are processed during this second pass. Finally, the object program and the listing files are written to a physical device.

c) What is the main difference between immediate and literal operands? *(3)*

- The data of an immediate operand is coded directly in the operand field of a machine instruction. A literal is expressed almost similarly than an immediate value in a symbolic statement but the assembler reserves implicitly a memory location for the literal value.

d) In what situation does the assembler generates refer and define records in the object program? *(2)*

- Refer and define records are generated when a routine refers to external symbols or defines external symbols. It is required that the routine explicitly specifies external references and definitions using the EXTREF and EXTDEF directives.

d) What is the main reason why a linking-loader requires two passes instead of one pass? *(2)*

- The process of the loader is similar to the assembler, during the first pass an external symbol table is built for the references that are linked from the external routines. During pass two, the loader updates the references in the linked program.

Question 2. *(25/60)*

Consider the following SIC/XE assembly language program.

	Sample	START	0	
	first	STL	retadr	172014
		LDCH	K30	532010
	loop	TD	output	E3 2006
		JEQ	loop	332FFA
		WD	output	DF2006
		LDL	retadr	082005
		RSUB		4C0000
	output	BYTE	X'05'	05
	k30	BYTE	C'0'	30
	retadr	RESW	1	

Concordia
University

Department of computer science
Tuesday, August 17 1999

Final examination

Instructor : Kerly Titus
Course : System software (comp229)
Room : H520

Supervisor : Kerly Titus
Section : CA
Time : 19:00-22:00

Allowed materials [Y] Calculator (non-programmable) [N] Documentation

Provided materials [Y] Addendum (see after questionaire)

Number of pages [6] (including cover page)

Special instructions : Return questionnaire.

Question 1. (SIC/XE architecture, assembler, loader) 30%

a) SIC/XE is downward compatible with SIC, and it can execute all the SIC instructions. However, SIC has one instruction format whereas SIC/XE has four different instruction formats. (6%)

- Which instruction format that SIC/XE uses to execute SIC instructions?
  - ***Format 3.***
- How does SIC/XE adapt the selected instruction format to be compatible with SIC?
  - ***Bits n and i are appended to the opcode and bits b, p, and e are appended to the address field.***

b) Explain two architectural features of a processor that make relocation bits more efficient than modification records to generate relocation information in an object program. (4%)

- ***Fixed instruction format.***

  Each relocation bit specifies the relative position of a memory reference that requires relocation within the text record. Therefore, it is necessary that the instruction format be fixed so that the operand reference be accessed.

- ***Direct addressing mode.***

  The only reason for using relocation is the use of direct memory references in the instructions. If there are no direct memory references, the relocation bits are not necessary and they use extra space.

c) Explain the main difference that exists between the program blocks and the control sections of the SIC/XE assembler. (6%)

- ***The program blocks and the control sections are both divisions of a main program and are handled separately by the assembler. However, the program blocks are logical divisions within the same source program whereas the control sections are independent divisions that use their own source and object programs. Also, control sections may in turn contain program blocks.***
  The program blocks are rearranged during pass 2 of the assembly process and the control sections are rearranged at link time.

d) The following code is an extract of the second pass algorithm for the linking-loader, and it processes modification records to perform relocation. A disadvantage of this code is that the external symbol table is sequentially searched for each reference of a symbol in a modification record. A more efficient code makes use of reference numbers instead of the symbol name in the modification record. (14%)

```
if (record_type == 'M')
{
  search(external_symbol_table, symbol_name)
  if (symbol_name is found)
     update memory location with value associated with symbol_name;
  else
     error_flag←undefined_external_symbol;
}
```

- Modify the pseudocode above so that it processes modification records that make use of reference numbers.

```
if (record_type == 'M')
{
   if (reference_table[modification_record.reference_number] == NULL)
   {
      search(external_symbol_table, reference_number);
      if (symbol is found)
         reference_table[reference_number] = symbol_address;
      else
      error_flag←undefined_external_symbol;
   }
   else
   update address field specified in modification record with numeric address from
   reference table;
}
```

- Explain the pros and cons of the new algorithm compared to the old version.

  - ***Pros.***

    - ***Faster search time to find the value of a symbol. Search time is of the order of 1 if a symbol has an entry in the reference table.***

  - ***Cons.***

    - ***Algorithm uses more codes.***
    - ***A new data structure (reference table) must be implemented to store the addresses of the symbols.***
    - ***The symbol table must include a new field to specify the reference number of a symbol.***

Question 2. (Compiler) 20%

a) Explain the two analysis steps that a compiler applies to a source program before it generates machine code for the language statements. (6%)

- *Lexical analysis (scanning).*

  - *The source program is scanned and the basic elements (tokens) of the programming language are recognized and classified.*

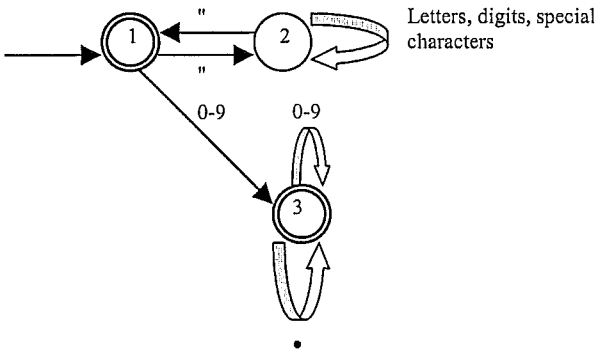
- *Syntactic analysis (parsing).*

  - *The tokens are analysed and a specific grammar rule is derived for each statement, where a statement is made up of the basic elements of the programming language. The analysis method generates a parse tree, and can be processed in a top-down or bottom-up approach.*

b) Show the new or modified rules for the simplified Pascal grammar (see addendum) if the REAL data type is added. (4%)

- *<type> ::= INTEGER | REAL*
  <factor> ::= id | int | real | (<exp>)

c) What are the valid strings that are accepted by the following finite automaton? To what elements of a programming language can you associate these valid strings? (8%)



- *Valid strings.*

  - *No input at all.*
  - *Sequence of characters enclosed between pairs of quotes.*
  - *Integer string of at least 1 digit.*
  - *Real string with at least1 digit before the decimal point.*
  - *Any sequence of digits and decimal points that starts with a digit.*

- *Programming language elements.*

  - *Character strings.*
  - *Integer values.*
  - *Real values.*

d) In the SIC/XE assembly language, the I/O operations are handled using the in-line statements TD, WR, and RD. However, the compiler generates out-of-line object code that calls the library routines XREAD or XWRITE to handle the I/O operations. (2%)

- What is the advantage to use the library routines XREAD and XWRITE?

  - ***The library routines reduce the number of statements in the object program.***

Question 3. (Process management) 20%

a) Explain what is really meant by the term "resource abstraction". Give an example. (5%)

- ***The resource abstraction is part of the system software and it provides a simple and general interface for the application programs to interact with the resources of the computer. Essentially, the physical and low-level characteristics of a specific device are hidden to the user.***
  The advantage of the resource abstraction is that the resources are more easily accessible, but some fine operations may not be processed adequately.

b) When a program requests execution from the operating system, is the resulting process immediately set in the running state and is allocated the processor resource? Explain briefly. (3%)

- ***If the operating system is monoprogramming the process is scheduled immediately. However, in a multiprogramming environment the process is placed on a ready queue to compete for the allocation of the processor with other processes. Also, the process must also compete to gain access to the resources that it may require during execution.***

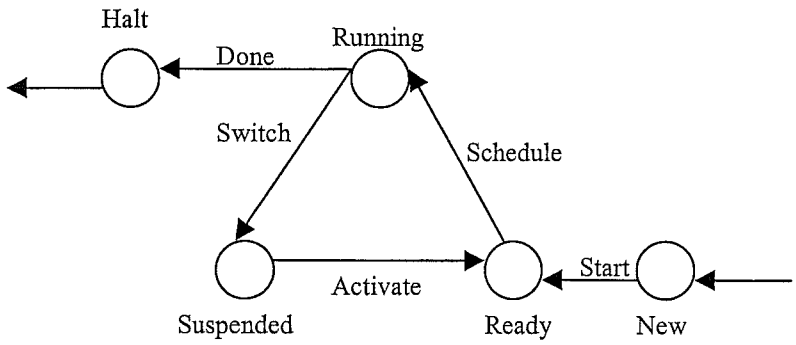
c) The computational environment is abstracted and is divided into user and system spaces. The user computations execute in the user space whereas the operating system (and system software) has exclusive access to the system (and user) space. However, user operations that require access to the system space must be handled by a special service request (system call or message) to the kernel of the operating system. (5%)

- Explain how the operating system processes the system calls.

  - ***The application software requests a specific service from the kernel by executing a system call routine.***
  - ***The operating system executes a trap instruction that changes the execution mode of the processor from user to system.***
  - ***The trap instruction branches to a branch table in order to locate the address of the routine that is required to service the system call.***
  - ***The system call routine is executed in system mode and at its completion control is returned to the operating system.***
  - ***The operating system switches the processor to the user mode and resumes the process that requested the system service.***

d) Suppose an operating system that requires a process to be allocated all its resource needs before it can execute, and therefore the process should not make any resource request in the running state. Also, a special synchronization mechanism allows the process manager to suspend a child process when the parent has been waiting for a threshold amount of time for the processor. In that case, the processor is switched between the parent and the child, and the child has to compete again for the processor. (7%)

- Draw the process state diagram for the scheduling stated above.



Question 4. (Memory management) 15%

a) It is stated that dynamic memory allocation is independent of the memory manager because the dynamic variables are simply bound to free addresses on the heap storage of the process. (4%)

- Does dynamic memory allocation never refers to the memory manager? Explain.

- ***When the heap storage of a process has been allocated, the dynamic memory allocation requests more memory from the memory manager. The address space of the process must be mapped to the new space allocated.***

b) Suppose that you have to design a memory management algorithm for a fixed-size memory. You have the option to implement a single queue where the processes wait for the best-fit partitions. The processes are inserted at the tail of the queue, and when a partition is released the first process waiting for that partition and nearest the head is allocated. (5%)

- What would be a disadvantage of the multiqueue implementation compared to the single queue approach?

- ***The multiqueue uses more space and it requires a little more overheads to manage it.***

c) Suppose a memory manager that implements the swapping strategy on a computer that does not provide supports for dynamic address relocation. (6%)

- Explain at what specific moment that the relocation of the memory operand references would take place.

- ***Relocation would take place at load time.***

- What major problem would occur when a program is relocated to a new memory location? Explain.

  - ***When the program is swapped out of memory, the memory operand references have been relocated and they must be changed when the program is swapped in to a new location.***
    A solution is to reload the object program with the modification records and to update the data that have changed. Thus, the memory manager must be aware of the location of the data area.

Question 5. (File management) 15%

a) Does the file manager maps low-level and structured files to storage device blocks using the same file operations? Explain. (4%)

- ***Low-level files are accessed at the byte level whereas structured files such as record-oriented and indexed sequential files are processed at the record level. Records and bytes are referenced by a relative position within the file but a record may contain multiple bytes. Thus, the file manager must use distinct functions to map low-level and structured files to the storage device blocks.***

b) Explain how an indexed-sequential file is processed? (5%)

- ***A primary key designating a search record is specified.***
- ***The index file is searched for that primary key.***
- ***The address of the record on the storage device is retrieved from the index file.***
- ***The record is accessed directly on the storage for a read or write operation.***
- ***The primary index key is validated against the primary key stored in the record, and if valid the record is retrieved and stored in memory.***

c) Consider a UNIX file structure that uses 12 direct, 1 single indirect, 1 double indirect, and 1 triple indirect block pointers. Suppose the block size is 1024 bytes and a block address requires 32 bits, what is the maximum size for a file? (6%)

- ***Number of addresses per disk block.***

  - $1024 / 4 = 256$

- ***Maximum file size.***

  - ***Size of direct pointers.***

    $12 \times 1024 = 12,288$ bytes.

- *Size of single indirect pointers.*

  256 ×1024 = 262, 144 bytes.

- *Size of double indirect pointers.*

  (256 ×256) ×1024 = 67, 108, 864 bytes.

- *Size of triple indirect pointers.*

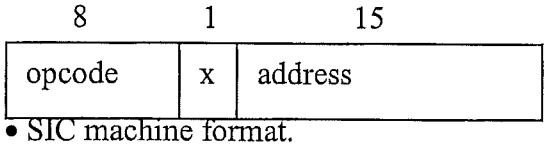
  (256 ×256 ×256) ×1024 = 17, 179, 869, 180 bytes.

- *Maximum size of the file.*

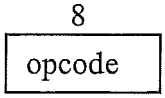
  17.247 252 480 tetrabytes.

Addendum

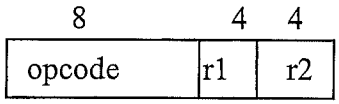
- **SIC & SIC/XE instruction formats.**



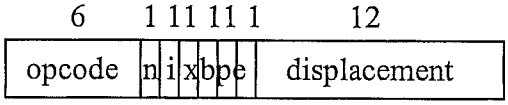
• SIC machine format.



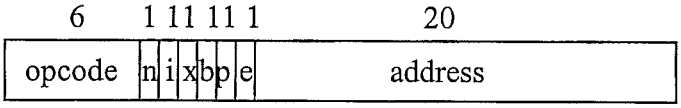
• SIC/XE – format 1.



• SIC/XE – format 2.



• SIC/XE – format 3.



• SIC/XE – format 4.

- **Pascal grammar in BNF.**

Rule number	Construct	Syntax description
1	<prog>	::=PROGRAM <prog-name> VAR <dec-list> BEGIN <stmt-list> END.
2	<prog-name>	::= id
3	<dec-list>	::= <dec> | <dec-list> ; <dec>
4	<dec>	::= <id-list> : <type>
5	<type>	::= INTEGER
6	<id-list>	::= id | <id-list> , id
7	<stmt-list>	::= <stmt> | <stmt-list> ; <stmt>
8	<stmt>	::= <assign> | <read> | <write> | <for>
9	<assign>	::= id := <exp>
10	<exp>	::= <term> | <exp> + <term> | <exp> - <term>
11	<term>	::= <factor> | <term> * <factor> | <term> DIV <factor>
12	<factor>	::= id | int | (<exp>)
13	<read>	::= READ (<id-list>)
14	<write>	::= WRITE (<id-list>)
15	<for>	::= FOR <index-exp> DO <body>
16	<index-exp>	::= id := <exp> TO <exp>
17	<body>	::= <stmt> | BEGIN <stmt-list> END

PART A

Answer ALL the Questions in this Part (3×12 = 36%)

1. Operating Systems

(a) [6%] Briefly discuss the major difference between batch processing and timesharing. When is batch processing the preferred strategy for work to be done by the computer? When is timesharing the preferred strategy? Explain.

(b) [6%] Suppose you want to implement a multiprogramming batch operating system on a computer that has no hardware interval timer. What problems might arise? Discuss a way to solve these problems using some other hardware or software mechanism.

2. Device Management

(a) [6%] What is an API (Application Programming Interface) and how is it related to device drivers? Explain why the UNIX device driver API uses operation names similar to the file interface (although they apply to physical devices rather than storage abstractions). What is the advantage and disadvantage of this approach?

(b) [6%] What is hardware buffering? How does it improve the I/O performance? For what kind of devices (character or block) is hardware buffering is more appropriate and why? Should disk controllers include hardware buffers? Explain your answer.

3. Memory Management

(a) [6%] Explain how virtual memory allows a process to use the CPU when only part of its address space is loaded in the primary memory. Why is program relocation unnecessary when virtual memory is used for memory management? Explain.

(b) [6%] Consider a computer system that implements swapping but does not have relocation hardware. Explain how a swapped out executable image is reloaded into memory in this system. Would it be possible for the swapping system to reload the data and stack segments? Explain how such a system might work, or why it would be impossible.

PART B

Answer ANY FOUR OF THE FIVE Questions in this Part (4×16 = 64%)

4. File Structure

(a) The UNIX file descriptor contains the following fields: Mode, UID, Group ID, Length in bytes, Length in blocks, Time of last modification, Time of last access, Time of last inode modification, Reference count, and Addresses of the data blocks.

(i) [4%] Notice that the filename is not in the file descriptor. Why is it not required to store the filename in the descriptor? Where is the filename stored and why?

(ii) [4%] What information is stored in the reference count field? Why is this information necessary and how is it used?

(iii) [4%] Explain the sequence of actions that take place when a *shared* file is closed by an application.

(b) [4%] What is the difference between indexed sequential file and a sequentially accessed file? Explain why a file system that supports indexed sequential files cannot be expected to have the same performance level as pure sequentially accessed files.

5. File Storage

(a) [4%] What is meant by disk fragmentation? What kind of fragmentation (internal or external) occurs in disks? Briefly discuss the similarities and differences between memory fragmentation and disk fragmentation.

(b) Consider a 10,000-byte file stored in a disk using 4096-byte blocks.

(i) [4%] How many blocks are used to store this file and how many bytes are wasted due to internal fragmentation (the unused number of bytes in the blocks)?

(ii) [4%] Suggest a simple way to reduce the number of bytes wasted due to internal fragmentation. What is the disadvantage of this approach?

(c) [4%] Consider a file that has just grown beyond its present space on disk (and thus requires one more disk block). Describe what steps will be taken next (to get the additional block) if the disk uses (i) contiguous allocation, (ii) linked allocation, and (iii) indexed allocation.

6. Assemblers

(a) [4%] What is a location counter? How is it used and where is it stored? Why there could be more than one location counters but only one program counter for a program? Explain.

(b) [4%] Explain how a symbol table is used during the assembly process. What kind of information does it contain? In what ways might the symbol table used by a compiler be different from the symbol table used by an assembler?

(c) Suppose that you are writing an assembler for a machine that has *only* program-counter relative addressing (that is, there are no direct-addressing instruction formats and no base relative addressing).

(i) [4%] Suppose that you wish to assemble an instruction whose operand is an absolute address in memory (for example, `LDA 100` to load register A from the hexadecimal address 100 in memory). How might such an instruction be assembled in a relocatable program? What relocation operations would be required?

(ii) [4%] Suppose that an instruction involving a forward reference is to be assembled. How might this be handled if the above assembler is a one-pass assembler?

7. Linkers, Loaders, and Program Libraries.

(a) [4%] Explain the fundamental difference between an linkage editor and a linking loader. When is it desirable to use a linkage editor instead of a linking loader? Explain.

(b) Dynamic linking is often used to allow several executing programs to share one copy of a subroutine or library.

(i) [4%] Briefly explain how dynamic linking allows run-time code sharing, mentioning how the dynamically linked subroutine is linked to the program calling it.

(ii) [4%] Suppose that routines that are brought into memory by dynamic linking need not be removed until the termination of the main program that uses them. Suggest a way to improve the efficiency of dynamic linking by making it unnecessary for the operating system to be involved in the transfer of control after the routine is loaded.

(iii) [4%] Suppose that it may be necessary to remove from memory routines that were dynamically loaded (to reuse the space). Will the method that you suggested in (ii) still work? What problems arise, and how might they be solved?

8. Compilers and Interpreters

(a) The following finite automaton recognizes identifiers containing the underscore character.

(i) [2%] What is the length of the shortest identifier recognized by the above finite automaton? Give an example.

(ii) [2%] Can the above finite automaton reject identifiers having more than 32 characters? If it can explain how; if it cannot, explain why not.

(b) [4%] A typical compiler has to do lexical analysis, syntactic analysis and code generation on a source program. Explain what processing is done in each of these steps. Which of these steps are machine independent and why?

(c) P-code compilers are very similar in concept to interpreters and produce platform independent code (that is, code that can be executed on different machines without any modifications).

(i) [4%] Explain how P-code compilers are conceptually similar to interpreters. What is the practical difference between the two?

(ii) [4%] Explain how the code compiled by a P-code compiler can be executed on different machines without any modifications.

*** ***END OF EXAMINATION*** ***

#27468516

CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

COMP 229, Sec. S Instructor: R. Jayakumar

MID-TERM EXAMINATION

Date: March 2, 1998 Time: 14:45–16:00

ANSWER ALL QUESTIONS

Concise Answers will be Appreciated

Please write your student number on the question paper and return it with the answer book

1. Operating Systems

(a) Processes and threads are two fundamental schedulable units of computation representing the execution of a program.

(i) [5%] What is the fundamental difference between processes and threads?

(ii) [8%] What is the difference between user threads and kernel threads? Can you implement user threads in an operating system that does not support kernel threads? Why or why not?

(b) Modern operating system requires the CPU to implement a mode bit so that the CPU can be run in either supervisor mode or user mode.

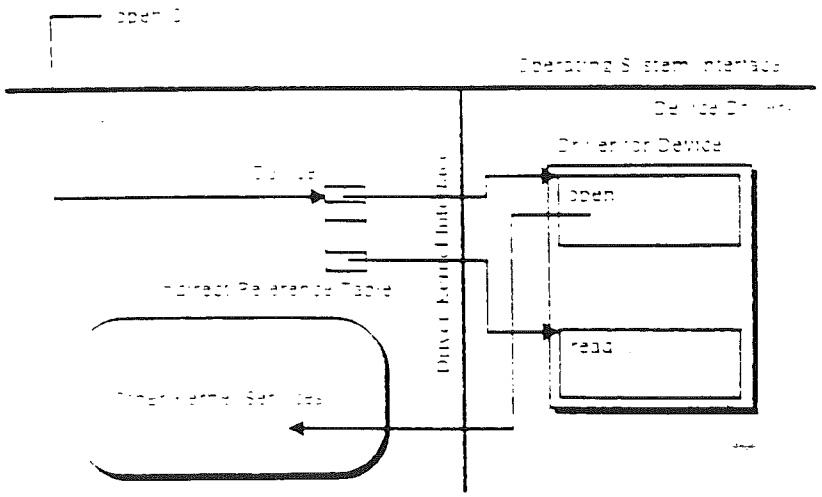
(i) [6%] Briefly explain how this mode bit is used to implement a secure operating system.

(ii) [6%] Can you design a secure operating system for a CPU which does not have the mode bit? If you can, explain how; otherwise, explain why not.

2. Device Management

(a) [10%] What is a reconfigurable device driver? Briefly explain why a reconfigurable device driver is better than the one directly compiled into the kernel.

(b) The following diagram shows the architecture of reconfigurable device drivers.



(i) [6%] Storage for a variable in a program may be allocated in data segment, in heap storage, or in the stack segment. Why is this necessary?

(ii) [7%] The heap storage and the stack segment share a common fixed block of memory and are growing towards each other. Why is this a better organization than using separate blocks of memory for the heap storage and for the stack segment? Explain.

(b) [12%] What is the difference between static relocation and dynamic relocation? Why is the relocation register required to support dynamic relocation? Can you write a dynamically relocatable program for a processor which does not have relocation register? Explain.

*** *END OF EXAMINATION* ***

I.D. #: 390125

CONCORDIA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

COMP 229, Sec. S — Instructor: R. Jayakumar

MID-TERM EXAMINATION

Date: March 2, 1998 — Time: 14:45–16:00

ANSWER ALL QUESTIONS

Concise Answers will be Appreciated

Please write your student number on the question paper and return it with the answer book

1. Operating Systems

(a) Processes and threads are two fundamental schedulable units of computation representing the execution of a program.

(i) [5%] What is the fundamental difference between processes and threads?

(ii) [8%] What is the difference between user threads and kernel threads? Can you implement user threads in an operating system that does not support kernel threads? Why or why not?

(b) Modern operating system requires the CPU to implement a mode bit so that the CPU can be run in either supervisor mode or user mode.

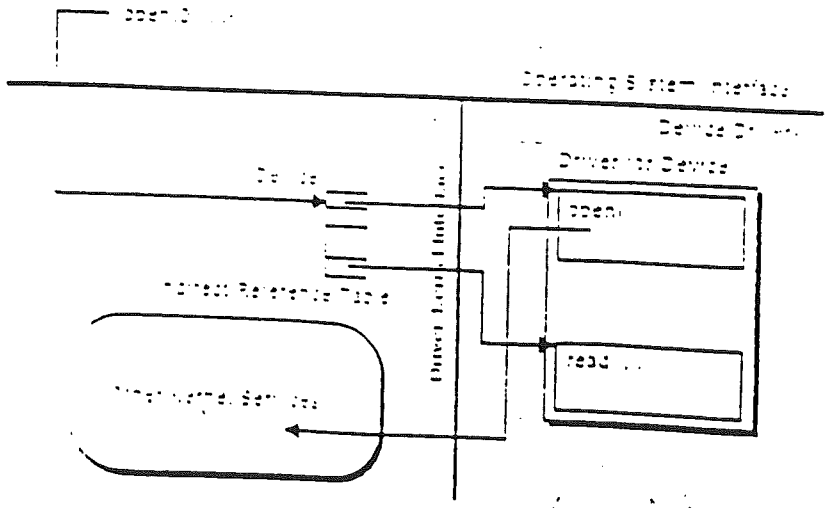
(i) [6%] Briefly explain how this mode bit is used to implement a secure operating system.

(ii) [6%] Can you design a secure operating system for a CPU which does not have the mode bit? If you can, explain how; otherwise, explain why not.

2. Device Management

(a) [10%] What is a reconfigurable device driver? Briefly explain why a reconfigurable device driver is better than the one directly compiled into the kernel.

(b) The following diagram shows the architecture of reconfigurable device drivers.



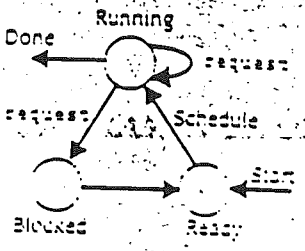
(i) [7%] Briefly explain the operating system interface and the driver-kernel interface.

(ii) [8%] Why is the indirect reference table required in a reconfigurable device driver? When and by whom is this table initialized?

3. Processor and Resource Management

(a) [10%] What are process descriptors? What kind of information process descriptors contain? Briefly explain how process descriptors are used in an operating system.

(b) The following is a simple state diagram for processes.



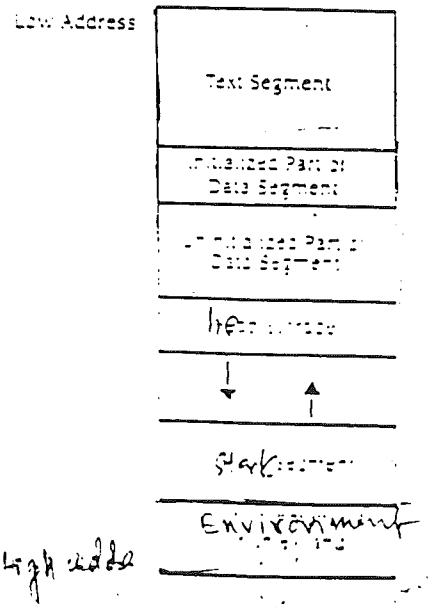
(i) [5%] Briefly explain the meaning of the three states (Running, Blocked, and Ready). When can a process be in each of these states?

(ii) [5%] Briefly explain the meaning of the two edges labeled request in this state diagram. When can a process change its state following each of these edges?

(iii) [5%] If you have enough resources in the system so that all requests are immediately allocated. How should the above state diagram be modified in this case? Explain.

4. Memory Management

(a) The following diagram shows UNIX-style memory layout.



CONCORDIA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

COMP 229, Sec. N — Instructor: R. Jayakumar

MID-TERM EXAMINATION

Date: October 23, 1998 — Time: 8:45–10:00

ANSWER ALL QUESTIONS

CLOSED-BOOK EXAMINATION

Concise Answers will be Appreciated

1. Assemblers

(a) [12%] Both location counter and program counter are used to keep track of the address of the next instruction in a program. Explain the difference between location counter and program counter. Why there could be more than one location counters but only one program counter for a program? Explain.

(b) [13%] Suppose that an instruction involving a forward reference is to be assembled using program-counter relative addressing. How might this be handled by an one-pass load-and-go assembler? Describe the data structures you would use and explain what happens when the instruction is assembled and when the referenced address is defined.

2. Linkers

(a) [12%] A linking loader usually makes two passes over its input. Suggest a design for a one-pass linking loader. What restrictions (if any) would be required? What would be the major advantage and the major disadvantage of such a one-pass loader?

(b) [13%] Explain the major difference between static program linking and dynamic program linking. Is it possible to use both static and dynamic linking in the same program? If so, explain why some program modules are statically linked and the others are dynamically linked. If not, explain why not.

3. Loaders

(a) [12%] Explain the major difference between a linkage editor and linking loader discussing the type of applications where each of them is desirable over the other. What type of loader (absolute, relocating, linking) does a system using a linkage editor need? Why?

(b) [13%] What is the difference between a bootstrap loader and a program loader? Why can a bootstrap loader be an absolute loader whereas a program loader is not? What kinds of errors might occur during bootstrap loading? What action should the bootstrap loader take for such errors?

4. Operating Systems

(a) [12%] Batch processing and timesharing are two strategies for executing programs by an operating system. What is the major difference between batch processing and timesharing? When is batch processing the preferred strategy for work to be done by the computer? When is timesharing the preferred strategy? Consider issues like improving system utilization and improving response time in your answer.

(b) [13%] What two advantages do user threads have over multiple processes? What major disadvantage do they have? Suggest the type of applications that would benefit from the use of threads.

*** ***END OF EXAMINATION*** ***