LSC Use Only No:	LSC Action-Date:	UWUCC USE Only No.	UWUCC Action-Date:	Senate Action Date:
07-166.			App-10/30/07	App-12/4/07

Curriculum Proposal Cover Sheet - University-Wide Undergraduate Curriculum Committee

Contact Person **James L. Wolfe**	Email Address **jlwolfe@iup.edu**
Proposing Department/Unit **Computer Science**	Phone **7-6104**

Check all appropriate lines and complete information as requested. Use a separate cover sheet for each course proposal and for each program proposal.

1. Course Proposals (check all that apply)

_____New Course _____Course Prefix Change _____Course Deletion

__X__Course Revision __X__Course Number and or Title Change __X__Catalog Description Change

COSC 300 Assembly Language	**COSC 300 Computer Organization & Assembly Language**
***Current* Course prefix, number and full title**	***Proposed* course prefix, number and full title, if changing**

2. Additional Course Designations: check if appropriate

_____ This course is also proposed as a Liberal Studies Course.

_____ This course is also proposed as an Honors College Course.

_____ Other: (e.g., Women's Studies, Pan-African)

3. Program Proposals

_____Catalog Description Change _____Program Revision

_____New Degree Program _____Program Title Change _____Other

_____New Minor Program _____New Track

***Current* program name**	***Proposed* program name, if changing**

4. Approvals		**Date**
Department Curriculum Committee Chair(s)	[signature]	5 Dec 06
Department Chair(s)	Wm. Ogilvie	12-7-06.
College Curriculum Committee Chair	[signature]	05/17/07
College Dean	Gerald Burch	9/24/07
Director of Liberal Studies *		
Director of Honors College *		
Provost *		
Additional signatures as appropriate: (include title)		
UWUCC Co-Chairs	Gail S Sechrist	10/30/07

* where applicable

Received
SEP 25 2007
Liberal Studies

Part II. Description of Curriculum Change

1. New Syllabus of Record

I. Catalog Description

COSC 300 Computer Organization & Assembly Language 3c-0l-3cr

Prerequisites: COSC 110 or equivalent

Discussion of the basic computer architecture elements: gates, combinational and sequential logic, hardware arithmetic, CPU and memory structure. Examination of the languages of machines: representation of data, addressing techniques, symbolic coding, assembly, and linking. Problem solving using assembly language.

II. Course Outcomes

Upon successful completion of this course, the student will be able to

A. Describe the operation of various logic gates and the theory (Boolean algebra) behind them.

B. Distinguish between combinational and sequential logic and discuss the function of the clock.

C. Describe how a CPU performs instructions during the fetch-decode-execute cycle and how the memory supports its actions.

D. Design simple digital logic to produce a specific result from given inputs and/or simplify digital logic to improve the efficiency of producing a result.

E. Describe how information of various data types are represented in a computer.

F. Explain how standard arithmetic operations (+, -, *, and /) are performed by the hardware.

G. Read, write, and debug programs in assembly language.

H. Explain the uses of various machine addressing modes and why they are used.

I. Use system services in a program.

J. Explain the internal workings of the machine on a procedure call and describe the structure of the call frame.

K. Distinguish between situations in which procedures or macros are appropriate.

III. Detailed Course Outline

A. Building logic gates from transistors 0.5 week
B. Application of Boolean algebra to digital logic 0.5 week
    1. Truth tables
    2. Identities
    3. Simplification of logic
C. Combinational logic 1.0 week
    1. Karnaugh Maps
    2. Quine-McKluskey algorithm
    3. Programmable Logic Arrays

D. Computer arithmetic — 1.0 week
    1. Half adder
    2. Adder-subtractor
        a. carry ripple adder
        b. carry lookahead adder
        c. fast adders
    3. Introduction to multiplication and division

E. Sequential logic — 1.0 week
    1. Flip flop types
    2. Role of the clock
    3. Use in registers

F. Memory hierarchy — 0.5 week
    1. Main memory organization
    2. Need for address translation
    3. Role of cache

G. CPU Organization — 1.0 week
    1. Clock
    2. Registers
    3. Fetch-decode-execute cycle
    4. Introduction to pipelined CPU

H. Number representation — 1.0 week
    1. 2's complement integers
        a. range of values
        b. overflow
    2. Floating point
        a. range of values (IEEE form)
        b. excess notation
        c. overflow and underflow
        d. special values
        e. accuracy, truncation, and roundoff
    3. Hexadecimal form

I. Assembly Language Forms — 1.0 week
    1. Instructions
        a. op codes
        b. operand forms
        c. addressing modes
    2. Directives
        a. reserving and initializing storage
        b. defining symbols
    3. Macros or pseudo instructions

J. Arithmetic instructions — 1.0 week
    1. Integer operations
    2. Floating point operations

K. Assembling, linking and executing assembly language programs — 0.5 week

L. Character manipulation instructions — 0.5 week

M. Bit operations — 0.5 week
    1. Shifting
    2. Logical operations

N. Conditional branching — 0.5 week
    1. Comparison of integers, floating point values, characters
    2. Limitations of branches
    3. Use of flag bits or registers

O. Loop Control — 0.5 week
    1. Development of loop algorithms
    2. Managing flags and distances

P. Conversion: numeric strings to/from 2's complement	0.5 week
Q. System services	0.5 week
1. Requesting input and output	
a. keyboard and screen	
b. files	
2. Getting information	
R. Procedures	1.0 week
1. Use of the stack	
2. Call frame	
3. Argument passing (by reference and by value)	
4. Recursion	
In-class examinations	1.0 week
Total	14 weeks
Final Exam	During Final Exam Week

IV. Evaluation Methods

55% Examinations. Three mid-term exams, each counting 12%; one final exam counting 19%. Examinations consist of short-answer, code segment, and analysis questions.

30% Assignments. Four to six programming assignments worth varying numbers of points, totaling 150-200 points. The assignments are directly related to the objectives.

15% Quizzes. Three or four announced quizzes on the lecture material.

Generally, assignments are expected to be handed in on time and examinations are expected to be taken on the designated date and time. Valid excuses, e.g. documented illness, are required for any exceptions to be made. Grading penalties for late assignments are rather severe: 20% for 1 day late, 40% for 2 days late, 70% for 3 days late, 100% after that. Quizzes cannot be made up; they must be taken when given. However, allowances can be made for valid excuses.

V. Example Grading Scale

90% - 100%	A
80% - 89%	B
70% - 79%	C
60% - 69%	D
below 60%	F

VI. Undergraduate Attendance Policy

Class attendance is regarded as being very important. Individual faculty may establish penalties for excessive numbers of unexcused absences. Excused absences will be allowed for illness, family emergencies, and involvement in university activities, such as sports. The penalties specified will meet university guidelines and be distributed to students with the course syllabus on the first day of class.

VII. Required Textbook, Supplemental Books and Readings

Irvine, K., *Assembly Language for Intel-Based Computers*, 5th edition, Prentice Hall, 2006.

An instructor supplement or one of these texts for the computer organization parts:
Mano, M. Morris, *Digital Design*, 3rd edition, Prentice Hall, 2001.

Stallings, W. *Computer Organization and Architecture*, 7th edition, Prentice Hall, 2005.

VIII. Special Resource Requirements

None.

IX. Bibliography

Abel, P., *IBM PC Assembly Language and Programming*, 5th edition, Prentice Hall, 2001.
Britton, R., *MIPS Assembly Language Programming*, Prentice Hall, 2003.

Carpinelli, J., *Computer Systems Organization and Architecture*, Addison Wesley, 2001.

Comer, D., *Essentials of Computer Architecture*, Prentice Hall, 2005.

Dunteman, J., *Assembly Language Step-by-Step*, Wiley, 2nd Ed., 2000.

Hamacher, C., Vranesic, Z. & Zaky, S., *Computer Organization*, 5th edition, McGraw Hill, 2002.

Irvine, K., *Assembly Language for Intel-Based Computers*, 5th edition, Prentice Hall, 2006.

Jones, W., *Assembly Language for the IBM PC Family*, 3rd edition, Scott/Jones Inc., 1997.

Juola, P., *Principles of Computer Organization And Assembly Language*, Prentice Hall, 2006.

Mano, M. M., *Digital Design*, 3rd edition, Prentice Hall, 2001.

Mazidi, M. & Mazidi, J., *80x86 IBM PC Compatible Computers, Assembly Language, Design and Interfacing, Volume I and II*, 4th edition, Prentice Hall, 2003.

Miller, K., *An Assembly Language Introduction to Computer Architecture*, Oxford, 1999.

Pamula, R., *Introduction to Computer Organization and Assembly Language Programming*, McGraw Hill, 1997.

Patt, Y. & Patel, S., *Introduction to Computing Systems: From Bits and Gates to C and Beyond*, McGraw Hill, 2001.

Patterson, D. & Hennessy, J., *Computer Organization and Design*, 3rd edition, Morgan Kaufmann, 2005.

Paul, R., *SPARC Architecture, Assembly Language Programming, and C*, 2nd edition, Prentice Hall, 1999.

Stallings, W., *Computer Organization and Architecture*, 7th edition, Prentice Hall, 2005.

Tanenbaum, A., *Structured Computer Organization*, 5th edition, Prentice Hall, 2006.

2. Summary of proposed revisions

The course title is being changed from COSC 300 Assembly Language to COSC 300 Computer Organization & Assembly language. The course content in the changed COSC 300 will devote about 50% of the course time to computer organization and the fundamentals of computer hardware; the remaining 50% will be spent on assembly language.

3. Justification

There are three motivating factors for this course change: the Computer Science GRE, ABET (Accreditation Board for Engineering and Technology) accreditation, and a perceived gap in student understanding of several fundamental concepts which is weakening the responses in senior-level classes.

For graduating Computer Science seniors who take the subject matter GRE, there is a clear difference between those who have taken COSC 410 Computer Architecture and those who have not. In the current curriculum, COSC 410 is the only course which specifically addresses issues related to computer organization and architecture. By moving the computer organization topics to COSC 300 (a core course), we will ensure that all Computer Science students learn the fundamentals of computer hardware and allow COSC 410 to concentrate on architecture and more advanced issues related to hardware.

Covering the fundamentals of computer organization in a core course will also help with ABET accreditation. ABET requires that this material be covered in an accredited program and, by having it in COSC 300, students are guaranteed of coverage.

Students who lack an understanding of hardware fundamentals struggle with several topic areas in COSC 432, Operating Systems, and COSC 424, Compiler Construction courses. COSC 300 is a prerequisite for both of these courses. Having an understanding of computer organization will allow students in either of these courses to improve what they do.

Little is lost from the current version of COSC 300 in making this change. All of the key areas of assembly language can still be covered. Several of the less-used elements can be eliminated; and the computer organization additions makes some assembly language topics easier to cover. There are likely to be fewer programming assignments, but added projects with digital logic.

4. Old Syllabus

I. Catalog Description

COSC 300 Assembly Language 3c-0l-3cr
Prerequisites: COSC 110 or equivalent

Examination of the structure and languages of machines; representation of data, addressing techniques, symbolic coding, assemblers, macros, etc.; problem solving using assembly language.

II. Course Objectives

Upon successful completion of this course, the student will be able to

A. Read, write, and debug programs written in assembly language.

B. Describe how information is represented with a computer.

C. Explain the uses of various machine addressing modes and why they are used.

D. Write assembly language macros to simplify the writing of large programs.

E. Describe how input and output at terminals and from files is performed at the machine level.

F. Use system services in a program.

G. Explain the internal workings of the machine on a procedure call and describe the structure of the call frame.

H. Discuss the relationship of assembly language and machine language and describe the linking process.

I. Explain the basic architecture of a computer with respect to storage of information and execution of instructions.

III. Course Outline

A. Introduction to Assembly Language (1 hr)
   1. Capabilities
   2. Machine dependence
   3. Beginning terminology

B. Machine Organization (2 hrs)
   1. Memory organization
   2. Bit and byte numbering
   3. CPU: registers
   4. Fetch-decode-execute cycle
   5. Condition flags

C. Assembly Language Forms (1 hr)
   1. Instructions

(op codes, operand forms, simple addressing modes)
2. Directives
(symbols and expressions)

D. Basic directives and instructions (1 hr)
1. Reserving and initializing storage
2. Defining symbols
3. Move, add, increment and branch instructions

E. Assembling, linking and debugging (1 hr)

F. Number representation (1 hr)
1. Sign-magnitude
2. 1's complement
3. 2's complement
4. Numeric conversions (decimal to hex and vice versa)

G. Integer arithmetic instructions (2 hrs)

H. Terminal input and output (1 hr)

I. Conditional branching (2 hrs)
1. Branch, compare and test instructions
2. Condition flags

J. Simple character manipulation (1 hr)

K. Conversion: numeric strings <-> 2's complement (2 hrs)
1. Horner's method
2. Type conversion instructions
3. Length conversion instructions

L. Loop Control (1 hr)

M. Terminal escape sequences (1 hr)

N. System services (3 hrs)

O. Assembly process (3 hrs)
1. Pass #1
(Location counter; symbol table)
2. Pass #2
(Generation of object code; operand forms; addressing modes; expression forms and evaluations)

P. Macros (5 hrs)
1. Definitions
2. Invocations
3. Position and Keyword forms
4. Local labels
5. User friendliness
6. Conditional assembly
7. Macro libraries

Q. Procedures (4 hrs)

1. Execution tasks
2. Internal subroutines
3. Psects and entries
4. CALLG and CALLS procedures
5. Call frame
6. Argument passing (by reference and by value)

R. File input and output (2 hrs)

S. Bit operations (2 hrs)
1. Masking
2. Shifting
3. Logical operations
4. Condition flags

T. Linking (1 hr)
1. Unresolved references
2. Object module
3. Address and displacement adjustments

U. Advanced character manipulation (1 hr)
1. Searching
2. Formatting numeric values

V. Floating point arithmetic (1 hr)

IV. Evaluation Methods

50% Examinations. Two mid-term exams, each counting 16%; one final exam counting 18%. Examinations consist of short-answer, code segment, and analysis questions.

40% Assignments. Six to eight programming assignments worth varying numbers of points, totaling 150-200 points. The assignments are directly related to the objectives.

10% Quizzes. Three or four announced quizzes on the lecture material.

Generally, assignments are expected to be handed in on time and examinations are expected to be taken on the designated date and time. Valid excuses, e.g. documented illness, are required for any exceptions to be made. Grading penalties for late assignments are rather severe: 20% for 1 day late, 40% for 2 days late, 70% for 3 days late, 100% after that. Quizzes cannot be made up; they must be taken when given. However, allowances can be made for valid excuses.

V. Required Textbook, Supplemental Books and Readings

Baase, Sarah, VAX-11 Assembly Language Programming, Prentice-Hall, 1992.

Various handouts based on computer system manuals.

VI. Special Resource Requirements

None.

VII. Bibliography

Sowell, E.F., Programming in Assembly Language VAX-11, Addison-Wesley, 1987.

Frank, T.S., Introduction to VAX-11 Architecture and Assembly Language, Prentice-Hall, 1987.

Brink & Spillman, Computer Architecture and VAX Assembly Language Programming, Benjamin/Cummings, 1987.

Kapps & Stafford, VAX Assembly Language and Architecture, PWS, 1985.

Pressman, M.H., Assembly Language Programming for the VAX-11, Mayfield, 1985.

Sebesta, R.W., Structured Assembly Language Programming, Benjamin/Cummings, 1984.

Part III. Letters of Support

Very few students from outside of Computer Science take this course. The majority are from Physics, Math, with a few from MIS. Letters of support are attached.

Subject: Re: COSC course revisions
Date: Fri, 8 Dec 2006 16:51:05 -0500
From: "hershman" <hershman@iup.edu>
To: "Jim Wolfe" <jlwolfe@iup.edu>

Jim,
The Physics Department recognizes the need of the Computer Science Department to revise its core course, COSC 300, in the manner they've described - namely to include Computer Organization as a major topic - and the rationale for doing so. This change should assist our physics majors in their preparation for our sister course to COSC 300 - PHYS 355, "Computer Interfacing",
Sincerely,
Ken Hershman
Chairman
IUP Physics Department.

----- Original Message -----
From: "Jim Wolfe" <jlwolfe@iup.edu>
To: "Gary Stoudt" <gsstoudt@iup.edu>; <hershman@iup.edu>; <woolcock@iup.edu>
Sent: Monday, September 18, 2006 9:42 AM
Subject: COSC course revisions

> *The Computer Science Department is revising several courses in its*
> *curriculum in preparation for applying for ABET accreditation. Two of*
> *those courses are COSC 250, Numerical Methods, and COSC 300, Computer*
> *Organization & Assembly Language. Because some students majoring in*
> *programs from your department (in addition to Computer Science majors)*
> *take these courses, we are asking you for a letter of support for the*
> *revision. The revision proposals are attached.*
>
> *Jim*

Subject: COSC 300 Support
Date: Thu, 7 Dec 2006 09:47:06 -0500
From: "Gary Stoudt" <Gary.Stoudt@iup.edu>
To: "'Jim Wolfe'" <jlwolfe@iup.edu>

Jim,

The Mathematics Department supports the revision to COSC 300. The changes would be beneficial to our majors who are minoring in COSC. The computer architecture topics and a de-emphasis of assembly language should give them a better understanding of computer fundamentals. Thank you for including us in your deliberations.

Gary

Gary Stoudt
Mathematics Department