LECTURE NOTES

I. INTRODUCTION

A. Welcome
B. Names
C. Schedule
D. Handouts
E. Why develop software for the Atari?
    1. better machine; you can do more.
    2. ultimately a bigger market (consumers)
    3. better support

II. PCS SYSTEM OVERVIEW

A. A very people-oriented computer
B. Hardware
    1. 6502 (1.79 MHz)
    2. RAM (192K possible)
    3. ROM (10K OS, 8K or 16K cartridge)
    4. VIA
    5. POKEY sound, controller ports, serial bus
    6. ANTIC display microprocessor
    7. CTIA television output
C. Large memory map:

```
 OS RAM   FMS   DUP  <-----user area-----> cartridge   hardware   firmware
---------I-----I-----I--------------------I-----------I----------I---------
  1.5K    5K    7K        up to 32K           8K          6K         10K
```

D. Strengths and weaknesses
    1. GRAPHICS!
    2. sound
    3. controllers
    4. 1.79 MHz 6502
    5. operating system (screen editing)
    6. BASIC variable names, indirection, syntax error handling

    7. BASIC strings
    8. slow disk
    9. little page zero or absolute RAM available
    10. maximum of five color registers in plain BASIC
    11. difficult to add hardware

HSCROL		XX		XX		XX		XX		XX		XX		XX		XX	Horizontal Scrolling
VSCROL			XX	XX			XX	XX			XX	XX			XX	XX	Vertical Scrolling
LD MEM SCAN					XX	XX	XX	XX					XX	XX	XX	XX	Load memory scan (3 byte)
INST INTERRUPT									XX	XX	XX	XX	XX	XX	XX	XX	Display instruction interrupt
BLK 1	00								80								Blank 1 line
" 2	10								90								Blank 2 lines
" 3-7																	Blank 3 thru 7 lines
" 8	70								F0								Blank 8 lines
JMP	01								81								Jump (3 byte instruction)
JVB	41								C1								Jump & wait for Vert. Blank
CHR (40,2,8)	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2	(also 3 byte)
" (40,2,10)	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3	
" (40,4,8)	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4	Character Mode
" (40,4,16)	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5	Instructions
" (20,5,8)	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6	
" (20,5,16)	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7	
MAP (40,4,8)	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8	
" (80,2,4)	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9	
" (80,4,4)	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA	
" (160,2,2)	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB	Memory Map Mode
" (160,2,1)	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC	Instructions
" (160,4,2)	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD	
" (160,4,1)	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE	
" (320,2,1)	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF	

Number of TV lines per cell
Number of Colors (Background + Playfield types)
Number of Horizontal cells (standard width screen)

Figure II.3 DISPLAY INSTRUCTION OPCODES

III. ANTIC AND THE DISPLAY LIST

A. Background: how a TV works; definitions of terms

B. display system used by other personal computers
   1. microprocessor-->RAM-->hardware-->screen
   2. the 'hardware' part is the limiting factor
   3. PET, TRS-80: fixed RAM, single mode (text). simple hardware
   4. Apple: two RAM sizes, 3 fundamental modes, better hardware

C. The Atari display system
   1. functional differences
      a. 14 fundamental display modes
      b. modes changeable on screen
      c. screen RAM anywhere
   2. ANTIC, a video microprocessor
   3. ANTIC's instruction set
      a. display instructions (graphics, text, or blank)
      b. JMP and JVB
      c. special options (scroll, DLI, and LMS)
   4. the display list (ANTIC'S program)
      a. synchronized to television cycle
   5. screen memory
   6. ANTIC writes only to CTIA

D. A sample display list (all values in hex)

```
70   Skip 8 lines
70   skip 8 lines
70   skip 8 lines
42   'mode 0' line with Load Memory Scan
LO   address of screen memory
HI   address of screen memory
xx   here follow mode bytes
xx   must total 192 horizontal scan lines
xx
xx
xx
xx
xx
xx
xx
41   Jump and wait for vertical blank...
LO   ...to beginning of display list
HI   high byte of address of display list
```

E. Using display lists
   1. make a paper image
   2. mode line setup
   3. scan line count
   4. arranging screen memory
      a. mixed mode screen memory discontinuities
      b. maintaining integer multiples
      c. brute force: calculating addresses
      d. setting up independent windows

F. Uses of display list manipulations
   1. vertical spacing
   2. mixed modes (note problems above!)
   3. access to new modes
   4. screen sequencing
   5. dynamic display lists

IV. INDIRECTION: COLOR REGISTERS AND CHARACTER SETS

A. Indirection is harder to understand but more powerful

B. Color register indirection
   1. One number in a color register controls the color of many pixels
   2. less RAM consumption
   3. more colors to choose from
   4. color cycling (animation)
   5. logically keying colors to situations
   6. display list interrupt capability

C. Character set indirection
   1. Direct: character code gives character in ROM
   2. Indirect: character code gives character in any specified table
   3. Procedure
      a. define characters on graph paper
      b. encode into bytes
      c. stuff into RAM
      d. POKE CHBAS with address of charset
      e. must start on 1K boundary (1/2K for GR. 1 & 2 sets)
      f. 8 bytes per character
   4. Capabilities
      a. multiple character sets (fonts)
      b. graphics character sets; maps; mixed text and graphics
      c. 4 color character sets
      d. vertical reflect
      e. time changes of character sets
      f. human slow: change of scenery
      g. human fast: animation
      h. machine fast: DLI's
   5. Overall assessment

V. PLAYER-MISSILE GRAPHICS

A. The problem: Animation

B. The traditional solution: playfield animation
   1. move bytes through RAM to move screen image
   2. address calculations; very slow
   3. resultant limitations:
      a. pure horizontal motion
      b. pure vertical motion
      c. few objects moving
      d. slow motion
   4. Essence of problem: 2d rasterized image, 1d RAM

C. The Atari solution: player-missile graphics
   1. Fundamental idea: 1d image, 1d RAM
   2. map table of bytes directly from RAM onto screen
   3. image is a vertical column 8 bits wide
   4. vertical motion with 1d move routine
   5. horizontal motion by horizontal position register

D. Embellishments
   1. 4 players, each with its own color register
   2. Controllable player widths
   3. Two vertical resolutions possible
   4. missiles (two bits wide)
   5. image priorities

E. How to do it
   1. Set aside player-missile RAM area (1K or 2K)
   2. set player colors
   3. set player widths and positions
   4. draw in players
   5. enable through PMBASE and DMACTL
   6. see demo program HOBBY1

F. Capabilities
   1. Animated objects
   2. Additional color
   3. Special offline characters
   4. cursors
   5. additional resolution by priority control

VI. DISPLAY LIST INTERRUPTS

A. A very powerful capability

B. Fundamental ideas
1. Screen drawing is time sequenced
2. By cutting into draw at appropriate time, image can be changed
3. change image by changing ANTIC registers
4. Timing provided by the DLI

C. How a working DLI happens
1. ANTIC encounters display list instruction with interrupt bit set
2. ANTIC checks NMIEN for enabling bit
3. If DLI is enabled, ANTIC interrupts 6502
4. 6502 vectors through $0200, $0201 to your DLI service routine
5. When done, 6502 resumes work.
6. Process repeats at same point on screen each 1/60th second.

D. How to set up a DLI
1. Write and place DLI service routine (page six?)
2. Set DLI bit in appropriate display list instruction
3. Set DLI vector in $0200, $0201
4. Enable DLI with $C0 into $D40E
5. Sample program: HOBBY2

E. Considerations
1. colors into ANTIC, not shadow. Attract
2. Execution time
3. things happening halfway across line; (WSYNC); keyboard IRQ
4. Multiple DLI's: separation problem, slow response
5. Using VCOUNT

F. Capabilities
1. Multiple playfield colors
2. Multiple color, width, position, and priority players
3. Multiple character sets; Rosetta Stone
4. * vertical screen architecture

VII. SCROLLING

A. Coarse scrolling
   1. method 1: move data through playfield with move routine.
   2. method 2: move playfield over data with LMS byte changes.
      a. serial scroll with single LMS instruction
      b. pure vertical scroll with single LMS instruction
      c. pure horizontal scroll with multiple LMS instructions
      d. mixing horizontal scroll with vertical scroll

B. Fine scrolling
   1. scrolls entire pixel line
   2. either horizontal or vertical or both
   3. line by line control
   4. DLI capability

C. How to do it
   1. set scroll bit in display list instruction
   2. store HSCROLL and VSCROLL values
   3. edge buffering
   4. coupling with coarse scrolling
   5. sample program: SCRL19.ASM

D. Applications
   1. Large images with screen window
      a. maps
      b. diagrams (schematics, blueprints)
      c. large blocks of text
      d. menus
      e. screen rotations (well, almost)